SIG MoBIS
Special Interest Group
on Modelling Business Information Systems

SIG EMISA
Special Interest Group
on Design Methods for Information Systems

# Enterprise Modelling and Information Systems Architectures

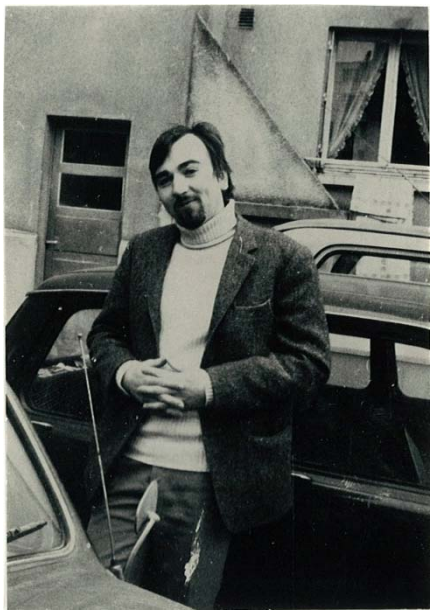
International Journal of Conceptual Modeling

**February 2018**

Special Issue on Conceptual Modelling in Honour of Heinrich C. Mayr

Edited by Judith Michael, Claudia Steinberger, Vladimir A. Shekhovtsov, Suneth Ranasinghe, Fadi Al Machot

# Table of Contents



***Heinrich C. Mayr***

*Picture sources: private, aau.at/hoi, aau.at/Maurer*

# Editorial Preface

This issue grew out of our desire to thank Heinrich C. Mayr for his various contributions to the scientific community. Thus, we dedicate this collection of inspiring articles a great boss, teacher, colleague and last but not least friend. His contributions have been celebrated at the UNISCON 2008 on occasion of his 60th birthday, in an official ceremony at the Alpen-Adria Universität Klagenfurt on occasion of his 65th birthday where he received the Golden Medal of the city of Klagenfurt, and at the INFORMATIK 2016 on occasion of his transition to emeritus status.

Heinrich's work always had a strong focus on human-centred informatics as an integral part of research. Moreover, Heinrich has made contributions to different areas of computer science: conceptual modelling for information systems, software and requirements engineering, computational linguistics, ontologies, model quality, context modelling and model centred architectures, all with a strong focus on user centredness and practical applications in areas as active assistance or medical domains.

All articles in this issue are related to conceptual modelling, an increasingly important topic in informatics and a main research area of Heinrich C. Mayr. As the ability of modelling is one of the fundamental cognitive skills of human beings and a method used in various domains up to and including our private life. Clearly, also conceptual modelling is used in a large variety of fields and the number of application areas is still growing.

We thank all authors and colleagues contributing to this issue and sharing their research ideas with us and the community. The articles range from Fundamentals of Conceptual Modelling, different modelling approaches like Petri Nets, Business Process Modelling, and Ontology-Based Modelling, Software and Requirements Engineering, Model Driven Engineering and Database Modelling to Conceptual Modelling and Applications.

We wish all readers great insights into challenging research topics in the area of conceptual modelling and inspiring ideas for further work.

Most of all, we would like to thank Heinrich. As a (Co-)Author and Editor of more than 210 scientific publications, supervisor of more than 100 master and diploma theses, 25 doctoral theses and several habilitation projects as well as program committee chairman and member of numerous international scientific conferences, he is a great role model: he conducts research passionate and on topics relevant to society in general, he always looks for practical applications of his theoretical approaches, and he was and is always willing to contribute to the international scientific community. Thank you for being inspiring, creative and passionate and that you have allowed us to walk a part of your way with you.

*Judith Michael*
*Claudia Steinberger*
*Vladimir A. Shekhovtsov*
*Suneth Ranasinghe*
*Fadi Al Machot*

***Guest Editors' contact information:***

**Dr. Judith Michael,**
**Dr. Claudia Steinberger,**
**Dr. Vladimir A. Shekhovtsov,**
**Suneth Ranasinghe**
Information Systems /
Application Engineering Group
Alpen-Adria-Universität Klagenfurt, Austria
judith.michael@aau.at
claudia.steinberger@aau.at
volodymyr.shekhovtsov@aau.at
suneth.ranasinghe@aau.at

**Dr. Fadi Al Machot**
Leibniz Center for Medicine and Biosciences
Research Center Borstel, Germany
falmachot@fz-borstel.de

# About Heinrich C. Mayr

Since this issue is dedicated to the 70th birthday of Heinrich C. Mayr, join us now on this tour through his life:

Heinrich was born on 1/2/48 – since this date is building a series whose terms are the successive powers of two, a perfect birthday for a computer scientist. Born in Miesbach, in the south of Germany, he grew up in Bavaria (Rosenheim and Munich) and moved to Karlsruhe, when his father was appointed there as a judge at the Federal court.

With regard to the main focus of his school education, it was not self-evident that he would become a professor of computer science: He attended a 'humanistic' Gymnasium (secondary school) which meant a broad education focussing on Ancient Greek and Latin. He also learned to play the violin, what inspired his love for music throughout his life. His interest in engineering, mainly in hydraulic engineering, became evident during his military service: He worked as a 'pioneer' and e. g. constructed bridges and ferry connections over the Danube. Back in civilian life again, reports and articles, announcing the first German Informatics degree program in Karlsruhe, arouse his particular interest in 1969.

Heinrich started studying Mathematics in summer 1969 at the Karlsruhe University and switched to Informatics in winter 1969. This year has also changed his private life: He met his future wife, Regina, participating in the first Informatics colloquium.

Working as a tutorial assistant for Prof. Nickel (Karlsruhe University), he got for the first time in touch with mainframes: he had to get results from UNIVAC computers using punched tapes. He completed his pre-degree in Karlsruhe in 1970 and decided to move to France for his further studies. As Informatics had already started in Grenoble in 1964, university's technical infrastructure in France was already further developed. He received a grant from the French government and moved to the University of Grenoble to finish his master study and to start with his PhD studies.

The main findings of his PhD thesis (written in French, 1975) under the title 'La Notion de demi-bande: étude algébrique et applications aux automates asynchrones' were published together with his supervisor Prof. Benzaken in the Semigroup Forum Vol. 10. In parallel, he already had returned to Karlsruhe, where he worked as a research and university assistant in 1973/74 for Prof. Schmitt and moved in 1975 to Prof. Lockemann. Their research focussed on database and information systems for large applications.

In 1978, he founded a software company as a spin-off together with Jürgen Kreutz where they developed a customizable ERP-System – using UNIX and hard disks of 25 MB. Their paradigms of providing software as a service for users, user friendly interfaces and highly customizable systems were successfully sold to wine, champagne and spirit producers. Subsequent versions of this software are still used by several customers and the company still exists. Heinrich himself sold his shares and left the company in 1991.

Despite his strong practical interests, he has never lost contact with the scientific community: Besides being a CEO for his company, he was still working for Prof. Lockemann in Karlsruhe. From 1983 to 1984 Heinrich worked as an associate professor at TU Munich in the research group of Prof. Bayer.

In 1990, Heinrich was appointed as a full professor for 'Praktische Informatik/Application Engineering' in Klagenfurt/Austria, where the study program 'Applied Informatics' had started in winter 1986. The emphasis on application orientation and a strong user focus has been an important concern for him since then. He supported the development and expansion of the study programme and was in 1994 founding dean of the Faculty of Economics and Informatics. He held

numerous academic positions at the Alpen-Adria-Universität Klagenfurt including being the rector from 2006 to 2012.

Beside university functions, he was member of the supervisory board of the Carinthian University College of Teacher Education, the supervisory board of the University for Applied Sciences in Carinthia and chairman of the supervisory board of Stadtwerke Klagenfurt (municipal service provider for energy, mobility and drinking water). However, he also had a broad engagement outside his University: in Informatics Societies, for Software Companies, for international conferences and for supporting Informatics in the Ukraine.

Having been a member of the German Informatics society GI e.V. since 1976, he funded in 1979, together with Bernd E. Meyer, the Special Interest Group on Design Methods for Information Systems (EMISA) becoming the chairman. Moreover, he was chairman of the TC 'Software Technology and Information Systems'. In 1996 he was elected as vice president of the GI and from 2000 to 2003 as president. Furthermore, he was vice president of CEPIS from 2004 to 2006. He funded the Lecture Notes in Informatics (LNI) series, where he is still Editor-in-Chief. For his engagement, he was appointed as a Fellow of the GI in 2008.

Over time, he established strong relations to Ukrainian universities. It started with the engagement of Prof. Schneider for economics and in 1993/1994 Heinrich started to support informatics scientists in Kherson and Kharkiv. He hosted and financed more than 30 master students and several visiting researchers. He was involved in the formation of the ISTA and UNISCON conference series and the introduction of the ECDL in the Ukraine. For his engagement he was appointed as Honoured Professor of Sciences of the Kherson State University in 2012 and has been holding a Doctor Honoris Causa from the National Technical University Kharkiv since 2001.

Heinrich has a talent for organizing events. Thus, he has organized several regional and international conferences, e. g. the INFORMATIK 1996 and 2016, Modellierung 2010, UNISCON 2008, or ER 2005. For his engagement for the ER community, he was named as an ER Fellow in 2013. Heinrich is member of numerous scientific committees, just to mention one, he is currently the chairman of the ER steering committee. Moreover, he was member of the board of trustees of the Klaus Tschira Stiftung in Heidelberg and he still serves as a Jury Member for the Software Engineering Award of the Ernst Denert-Stiftung since 1997.

Heinrich's work was always strongly related to applications of informatics in practice. Since 2016 he has established the university level program 'IT Business Solutions' to establish stronger relations to companies and enhance the lifelong learning of IT personnel. In direct contact to IT companies he was related in the establishment of the SIC (Software Internet Cluster), where he was vice president and is currently chairman of the council. In 2016, he received the grand medal of honour of the Carinthian chamber of commerce for his engagement for software companies in the region.

During the last decade, he and his team (to which we were allowed to belong) focused on human centred research in model engineering, design and realization of user-oriented application architectures with a strong focus on the integration of user needs into the development process and customer centred, accessible, effective and sustainable software and services. He and his team conducted both fundamental research as well as experimental and applied research in close cooperation with industry following the principle: 'There is nothing more practical than a good theory' (see https://ae-ainf.aau.at/).

*On a personal level, Heinrich is a very social person who believes in the goodness in people. His family plays an important role in his life. To do research is for him both, work and one of his most beloved hobbies. We are proud to have been part of his team.*

**Judith Michael**
**Claudia Steinberger**

*February 2018*

# Conceptual Model Notions – A Matter of Controversy

## Conceptual Modelling and its Lacunas

Bernhard Thalheim*,a

[a] Department of Computer Science, Christian Albrechts University Kiel, Germany

Abstract. *The conception of a conceptual model is differently defined in Computer Science and Engineering as well as in other sciences. There is no common notion of this conception yet. The same is valid for the understanding of the notion of model. One notion is:* A model is a well-formed, adequate, and dependable instrument that represents origins and functions in some utilisation scenario. *The* conceptual model of an information system *consists of a conceptual schema and of a collection of conceptual views that are associated (in most cases tightly by a mapping facility) to the conceptual schema. In a nutshell, a* conceptual model *is an enhancement of a model by concepts from a concept(ion) space.*
*The variety of notions for conceptual model is rather broad. We analyse some of the notions, systematise these notions, and discuss essential ingredients of conceptual models. This discussion allows to derive a research program in our area.*

Keywords. Model • Conceptual Model • Concept and Notion of a Model • Art of Modelling

## 1 What is a Conceptual Model

Modelling is a topic that has already been in the centre of research in computer engineering and computer science since its beginnings. It is an old sub-discipline of most natural sciences with a history of more than 2.500 years. It is often restricted to Mathematics and mathematical models what is however to much limiting the focus and the scope. Meanwhile it became a branch in the Philosophy of Science. The number of papers devoted to modelling doubles each year since the early 2000's.

It is often claimed that there cannot be a common notion of model that can be used in sciences, engineering, and daily life. The following notion covers all known so far notions in agriculture, archaeology, arts, biology, chemistry, computer science, economics, electrotechnics, environmental sciences, farming, geosciences, historical sciences, languages, mathematics, medicine, ocean sciences, pedagogical science, philosophy, physics, political sciences, sociology, and sports. The models used in these disciplines are instruments that are deployed in certain scenarios (see Thalheim and Nissen 2015). A commonly acceptable statement for a general model notion is the following one[1]:
A model is a *well-formed*, *adequate*, and *dependable* instrument that represents *origins* and *functions* in some utilisation *scenario*. Its criteria of well-formedness, adequacy, and dependability must be commonly accepted by its *community of practice* within some *context* and correspond to the *functions* that a model fulfils in *utilisation scenarios*. The function determines the purposes and goals.

CS-conceptual modelling[2] is often related back to the introduction of the entity-relationship

* Corresponding author.
E-mail. thalheim@is.informatik.uni-kiel.de

[1] We refer to the model-to_model-modelling compendium (see Thalheim and Nissen 2015) for notions that are not introduced in this paper.

[2] In the paper we restrict ourselves to this kind of conceptual model and thus omit the CS acronym. In general, a conceptual model is a representation of a system in its widest sense on

model(ling language) for information systems development. It surprises nowadays that there is no commonly accepted notion of conceptual model yet. There have been several trials but none of them was sufficient and was able to cover the idea of the conceptual model.

The database and information systems research communities are extensively using the term "conceptual model"[3]. The notion of conceptual model still needs some clarification: what is a conceptual model and what not; which application scenario use which kind of conceptual model; is conceptual modelling only database modelling; do we need to have an understanding of modelling; is a conceptual database model only a reflection of a logical database model; is a conceptual model a model or not; etc. Let us illustrate the wide spread and understanding of conceptual models, the activity of conceptual modelling, and the modelling as a scientific and engineering process by some examples[4,5]:

*Reality and world description:* Conceptual modelling is the activity of formally describing some aspects of the physical and social world around us for purposes of understanding and communication. Such descriptions, often referred as conceptual schemata, require the adoption of a formal notation, a conceptual model in our terminology[6]. (see Mylopoulos 1992)

*Community description :* Conceptual modelling is about describing the semantics of software applications at a high level of abstraction[7]. Specifically, conceptual modellers (1) describe structure models in terms of entities, relationships, and constraints; (2) describe behaviour or functional models in terms of states, transitions among states, and actions performed in states and transitions; and (3) describe interactions and user interfaces in terms of messages sent and received and information exchanged. In their typical usage, conceptual-model diagrams are high-level abstractions that enable clients and analysts to understand one another, enable analysts to communicate successfully with application programmers, and in some cases automatically generate (parts of) the software application. (see ER community 2017)

*Conceptual database modelling:* A data model is a collection of concepts that can be used to describe a set of data and operations to manipulate the data. When a data model describes a set of concepts from a given reality, we call it a conceptual model. (see Batini et al. 1992; Elmasri and Navathe 2000[8])

*Instance-integrating conceptual modelling:* A conceptual model consists of a conceptual schema

the basis of concept(ion)s that allow people to consciously act and being guided in certain situations of their systems.

[3] Facetted search for the term "conceptual model" in DBLP results in more than 5.000 hits for titles in papers (normal DBLP search also above 3.400 titles).

[4] The notion of conceptualisation, conceptual models, and concepts are far older than considered in computer science. The earliest contribution to models and conceptualisations we are aware of is pre-socratic philosophy.

[5] Wikiquote (see Wikiquote 2017) lists almost 40 notions. We add our list to this list.

[6] And continuing: These terms are introduced by analogy to data models and database schemata. The reader may want to think of data models as special conceptual models where the intended matter consists of data structures and associated operations.

[7] Some research challenges in conceptual modelling: Provide the right set of modelling constructs at the right level of abstraction to enable successfully communication among clients, analysts, and application programmers. Formalize conceptual-modelling abstractions so that they retain their ease-of-communication property and yet are able to (partially or even fully) generate functioning application software. Make conceptual modelling serve as analysis and development tools for exotic applications such as: modelling the computational features of DNA-level life to improve human genome understanding, annotating text conceptually in order to superimpose a web of knowledge over document collections, leveraging conceptual models to integrate data (virtually or actually) providing users with a unified view of a collection of data, extending conceptual-modelling to support geometric and spatial modelling, and managing the evolution and migration information systems. Develop a theory of conceptual models and conceptual modelling and establish a formal foundation of conceptual modelling.

[8] Another version is the following one: The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. A conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. A high-level data model or an implementation data model can be used at this level.

and an information base. A conceptual schema provides a language for reasoning about an object system, and it specifies rules for the structure and the behaviour of the system. A description of a particular state is given in an information base, which is a set of type and attribute statements expressed in the language of the conceptual schema. (see Boman et al. 1997)

*System-representation models:* A conceptual model is a descriptive model of a system based on qualitative assumptions about its elements, their interrelationships, and system boundaries. (see Business dictionary 2017)

*Representational models:* A conceptual model is a type of diagram which shows of a set of relationships between factors that are believed to impact or lead to a target condition; a diagram that defines theoretical entities, objects, or conditions of a system and the relationships between them. (see WordNet dictionary 2017)

*Enterprise modelling and conceptual modelling:* A conceptual is a model which represents a conceptual understanding (i. e. conceptualisation) of some domain for a particular purpose. A model is an artefact acknowledged by the observer as representing some domain for a particular purpose. (see Bjeković 2017)

*Holistic view:* In most cases, a model is also a conceptual model [9]. (see Pastor 2016)

*Conceptual models as a result of an activity:* We use the name of conceptual modelling for the activity that elicits and describes general knowledge a particular information system needs to know. The main objective of conceptual modeling is to obtain that description, which is called a conceptual schema. (see Olivé 2007)

*Purpose-oriented modelling:* Conceptual modelling is about abstracting a model that is fit-for-purpose and by this we mean a model that is valid, credible, feasible and useful. (see Robinson 2010)

*Documentation-oriented conceptual model:* A conceptual data model is a summary-level data model that is most often used on strategic data projects. It typically describes an entire enterprise. Due to its highly abstract nature, it may be referred to as a conceptual model. (see InfoAdvisors 2017)

*Semiotics viewpoint:* Conceptual modelling is about describing syntax, and semantics (potentially also pragmatics) of software applications at a high level of abstraction. (see Embley and Thalheim 2011)

*Documentation and understanding viewpoint:* A conceptual model of an application is the model of the application that the designers want users to understand. By using the application, talking with other users, and reading the documentation, users build a model in their minds of how to use the application. Hopefully, the model that users build in their minds is close to the one the designers intended. (see Johnson and Henderson 2013)

*Conceptualisations of models:* Conceptual models are nothing else as models that incorporate concepts and conceptions which are denoted by names in a given name space. A concept space [10] consists of concepts (see Murphy 2001) as basic elements, constructors for inductive construction of complex elements called conceptions, a number of relations among elements that satisfy a number of axioms, and functions defined on elements. (see Thalheim 2017)

At the ER'2017 conference a special brainstorming and discussion session has been organised with the task to coin the notion of a conceptual model. It seems to be surprising that there is no commonly accepted notion of a conceptual model after more than 40 years of introduction of this concept into database research. One proposal of the brainstorming discussion was:

[9] The slides of the keynote talk state: A conceptual model is a simplification of a system built with an intended goal in mind.
An abstraction of a system to reason about it (either a physical system or a real or language-based system). A description of specification of a system and its environment for some purpose. One main conclusion that we can reach is that the distinction between "model" and "conceptual model" is not always as precise as it should be.

[10] We follow R.T. White (see Thalheim 2014; White 1994) and distinguish between concepts, conceptual, conceptional, and conceptions.

*ER 2017 discussion proposal:* A conceptual model is a partial representation of a domain that can answer a question.

As for a model, the purpose dimension determines the quality characteristics and the properties of a model.

In a nutshell, a *conceptual model* is an enhancement of a model by concepts from a concept(ion) space. It is formulated in a language that allows well-structured formulations, is based on mental/perception/domain-situation models with their embedded concept(ion)s, and is oriented on a modelling matrix that is a common consensus within its community of practice.

We thus meet a good number of challenges, e.g. the following ones: is there any acceptable and general notion of conceptual model; do conceptual models really provide an added and sustainable value; what are the differences between conceptual models and models; what is a model; what means conceptualisation; how to support language-based conceptual modelling; etc. This paper is oriented on these questions and tries to develop an answer to them. We restrict the investigation to conceptual models in computer science and computer engineering and thus do not consider conceptual modelling for product design, service design, other system's design, natural and social sciences. Physical conceptual models are also left out of scope.

## 2 Revisiting Conceptual Modelling

### 2.1 State-Of-Art and State-Of-Needs

Modelling offers the benefit of producing better and understandable systems. It is based on a higher level of abstraction compared to most programming languages. Whether a model must be formal is an open question. The best approach is to consider model suites (or ensembles) that consist of a coherent collection of models which are representing different points of view and attention. We observe a resurgence in domain specific approaches that are challenged by technical, organisational and especially language design problems. UML is not the solution yet because UML Models are not executable but MDA needs them to be. The vast majority of UML models we have seen in industrial project are mere sketches and are informal and incomplete. They are not yet a viable basis for precise and executable models. Without precise models, no formal checking can take place. Therefore, these issues must be addressed either if modelling is well-accepted and gains significant presence in applications.

From the other side, the large body of knowledge on conceptual modelling in computer science is a results of hundreds of research papers over the last three-score years, although different names have been used for it. Modelling is often based on a finalised-model-of-the-real-world paradigm despite the constant change in applications. Model quality has already been considered in a dozen papers. Modelling literacy is rarely addressed in education. Models must however be reliable, refinable, and translatable artefacts in software processes.

Conceptual modelling is supported by a large variety of tools. e.g. (see Karagiannis et al. 2016). However, few of them support executable models. Of that few, far fewer still are actually rewarding to use. Conceptual models are acknowledged as mediators in the software development process. However, they are used and then not evolving with the evolution of the software. Reuse, migration, adaptation, and integration of models is still a lacuna. The lack of robust, evolution-prone and convenient translators is one reason. An environment as a constituent part for modelling and translation into consistent, easy-to-use and -revise, seamless, and industry-quality tool is still on the agenda. Information and software systems become eco-systems. Modelling eco-systems are not properly addressed yet.

Models are also used for communication based on some injection of a name space, while the community of practice uses a wealth of terms and terminology, with which they express their nuances of viewpoints. So, we need a number of representation models beside the singleton graphical representation. At the same time, models must

be properly formal and based on rules strictly to be followed or else having the risk of making illogical statements. Thus, modelling must be based on methodologies.

## 2.2 Myths of (Conceptual) Modelling

Modelling and especially conceptual modelling is not well understood yet and misinterpreted in a variety of ways. It has brought a good number of myths similar to those known for software development (see Ambler and Sadalage 2006):

1. *Modelling is mainly for documentation*. The introduction of conceptual modelling for database systems has been motivated by a documentation scenario. A conclusion might be that modelling is a superfluous activity, especially in the case that documentation is not an issue.
2. *Modelling is finished with the use of the model and an initial phase*. Historic development of software started with requirements which were frozen afterwards and with modelling and specifications that were complete and became frozen before realisation begins.
3. *Modelling is only useful for heavyweight V-style software development*. Modelling and especially conceptual modelling is abandoned due to its burden and the discovery of the complexity of the software that is targeted.
4. *The collection of origins must be "frozen" before starting with modelling*. Models should be plastic and stable (one of the justification and thus dependability properties), i. e. the collection of origins to be modelled could change.
5. *The model is carved in stone and changes only from time to time if at all*. The realisation becomes 'alive' and thus meets continuous change requests. The model can have some faults, errors, misconceptions, misses etc. Extensions and additional services are common for systems. So, the model has to change as well.
6. *Modelling starts with selecting and accommodating a CASE tool*. Although CASE tools are useful, they impose their own philosophy, language, and treatment. Moreover, CASE tools allow to become too detailed. Instead, conceptual modelling should allow to create the model that is simple as possible and as detailed as necessary.
7. *Conceptual modelling is a waste of time*. Developers are interested in quick success and have their own perception model in mind. It seems to be superfluous to model and better to focus solely on how to write the code.
8. *Conceptual data modelling is a primary concern*. Data- and structure-driven development without consideration of the usage of the data in applications results in 'optimal' or 'normalised' data structure models and bad database performance. One must keep in mind the usage of the data, i. e. use a co-design method, e.g. (see Thalheim 2000).
9. *The community of practice has a common understanding how to conceptually model*. Modelling skills evolve over years and are based on modelling practice and experience. Further, conceptual models are based on a common domain-situation model that has to be shared within the community of practice. So, the perception models of modellers should match.
10. *Modelling is independent on the language*. Modelling cannot be performed in any language environment. Language matters, enables, restricts and biases (see Whorf 1980).

Understanding these and other myths allows to better understand the modelling process and the models. One way to overcome them is the development of sophisticated and acknowledged frameworks. Model-centred development (see Mayr et al. 2017) uses models as a kernel for the development of systems. Conceptual modelling is still taught as modelling in the small whereas modelling in the large is the real challenge.

## 2.3 Specifics of Notions

Let us return to the list of notions given in Section 1. Each of these notions has its graces, biases, orientations, applicability, acceptability, and specifics.

*Scopes of conceptual models* may vary from very general models to fine-grained models. General models allow to reason on system properties whereas fine-grained models serve as a blueprint for development.

*Result-oriented viewpoint:* Conceptual models can be seen as the final result and documentation of an activity that follows a certain development strategy such as agile, extreme, waterfall etc. methodologies.

*Communication viewpoint:* Conceptual models are a means for communication and negotiation among different stakeholders.

*System construction orientation:* Database, information and software system development is becoming more complex, more voluminous, requires higher variety, and changes with higher velocity. So a quick and parsimonious comprehension becomes essential and supports higher veracity and an added value for the system itself.

*Perception and domain-situation models* are specific mental models either of one member or of the community of practice within one application area. It is not the real world or the reality what is represented. It is the common consensus, world view and perception what is represented.

*Conceptual models as documentation:* Models provide also quality in use, i. e. they allow to survey, to understand, to negotiate, and to communicate.

*Conceptual modelling with prototypes:* Models can be enhanced by prototypes or sample populations. A typical approach is sample-based development (see Halpin 2009).

*Visualisation issues:* Conceptual models may be combined with representation models, e.g. visualisation models on the basis of diagrammatic languages.

*Biased conceptual modelling approaches:* Conceptual models are often models with a hidden background, especially hidden assumptions, that are commonly accepted in a community of practice in a given context and utilisation scenario.

*Semiotics and semiology of conceptual modelling:* Conceptual models are often language-based. The language selection is predetermined and not a matter of consideration in the modelling process.

*Quality models:* Conceptual models should be well-formed and satisfy quality requirements depending on their function in utilisation scenarios.

*Concepts, conceptions:* The elements in a conceptual models are annotated by names from some name space. These names provide a reference to the meaning, i. e. a reference to concepts and conceptions in a concept space.

*Conceptual model suites:* Models can be holistic or consist of several associated models where in the latter case each of them represents different viewpoints. For instance, a conceptual database model consists of a schema and a number of derived views which represent viewpoints of business users.

*Normal models:* Conceptual models represent only certain aspects and are considered to be intentionally enhanced by elements that stem from common sense, consensuses, and contexts.

A normal model (called 'lumped' model in Zeigler et al. 2000) is a part of the model that is considered to be essential and absolutely necessary. The *normal model* has a context, a community of practice that puts up with it, a utilisation scenario for which is is minimally sufficient, and a latent – or better deep – model on which it is based (see Zeigler et al. 2000 for 'base' model). The *deep model* combines the unchangeable part of a model and is determined by the grounding for modelling (paradigms, postulates, restrictions, theories, culture, foundations, conventions, authorities), the outer directives (context and community of practice), and the basis (assumptions, general concept space, practices, language as carrier, thought community and thought style, methodology, pattern,

routines, common sense) of modelling. The (modelling) *matrix* consists of the deep model and the modelling scenarios. The last ones are typically stereotyped in dependence on the chosen *modelling method*.

This variety of viewpoints to conceptual models illustrates the different requirements and objectives of models. So, we might ask whether a common notion of a conceptual model exists or whether we should use different notions.

## 2.4 Problems and Challenges

Conceptual modelling techniques suffer from a number of weaknesses. These weaknesses are are mainly caused by concentration on database modelling and by non-consideration of application domain problems that must be solved by information systems. We follow the state-of-the-art analysis of A. van Lamsweerde (see van Lamsweerde 2000, 2008) who gave a critical insight into software specification and arrive with the following general weaknesses for conceptual modelling of information and database systems:

*Limited scope.* The vast majority of techniques are limited to the specification of data structuring, that is, properties about what the schema of the database system is expected to do. Classical functional and non-functional properties are in general left outside or delayed until coding.

*Poor separation of concerns.* Most modelling approaches provide no support for making a clear separation between (a) intended properties of the system considered, (b) assumptions about the environment of this system, and (c) properties of the application domain

*Low-level schematology.* The concepts in terms of which problems have to be structured and formalized are concepts of modelling in the small - most often, data types and some operations. It is time to raise the level of abstraction and conceptual richness found in application domains.

*Isolation.* Database modelling approaches are isolated from other software products and processes both vertically and horizontally. They neither pay attention to what upstream products in the software might provide or require nor pay attention to what companion products should support nor provide a link to application domain description.

*Poor guidance.* The main emphasis in database modelling literature has been on suitable sets of notations and on a posteriori analysis of database schemata written using such notations. Constructive methods for building correct models for complex database or information systems in a safe, systematic, incremental way are by and large non-existent.

*Cost.* Many information systems modelling approaches require high expertise in database systems and in the white-box use of tools.

*Poor tool feedback.* Many database system development tools are effective at pointing out problems, but in general they do a poor job of (a) suggesting causes at the root of such problems, and (b) proposing better modelling solutions.

Modern modelling approaches must not start from scratch. We can reuse achievements of database modelling in a systematic form and thus maintain theories and technologies while supporting new paradigms.

*Constructiveness.* Models of information systems can be built incrementally from higher-level ones in a way that guarantees high quality by construction. A method, is typically made of a collection of model building strategies, paradigm and high-level solution selection rules, model refinement rules, guidelines, and heuristics. Some of them might be domain-independent, some others might be domain-specific.

*Support for comparative analysis.* Database models depend on the experience of the developer, the background or reference solutions on hand, and on preferences of developers. Therefore, the results within a team of developers might need a revision or a transformation to a holistic solution. Beyond modelling qualities we

may develop precise criteria and measures for assessing models and comparing their relative merits.

*Integration.* Tomorrow's modelling should care for the vertical and horizontal integration of models within the entire analysis, design, development, deployment and maintenance life cycle - from high-level goals to be supported by appropriate architectures, from informal formulation of information system models to conceptual models, and from conceptual models to implementation models and their integration into the deployment of information systems.

*Higher level of abstraction.* Information systems modelling should move from infological design to holistic co-design of structuring, functionality, interactivity and distribution. These techniques must additionally be error-prone due to the complexity of modern information systems. These abstraction techniques may be combined with refinement techniques similar to those that have been developed for abstract state machines.

*Richer structuring mechanisms.* Most modelling paradigms of the modelling-in-the-small approach, available so far for modularising large database schemata, have been lifted from software engineering approaches, e. g., component development. Problem-oriented constructs are developed as well as model suites that provide a means for handling a variety of models and viewpoints.

*Extended scope.* Information system development approaches need to be extended in order to cope with the co-design of structuring, functionality, interactivity and distribution despite an explicit treatment of quality or non-functional properties.

*Separation of concerns.* Information system modelling languages should enforce a strict separation between descriptive and prescriptive properties, to be exploited by analysis tools accordingly.

*Lightweight techniques.* The use of novel modelling paradigms should not require deep theoretical background or a deep insight into information systems technology. The results or models should be compiled to appropriate implementations.

*Multi-paradigm modelling.* Complex information systems have multiple facets. Since no single modelling paradigm or universal language will ever serve all purposes of a system. The various facets then need to be linked to each other in a coherent way.

*Multilevel reasoning and analysis.* A multi-paradigm framework should support different levels of modelling, analysis, design and development - from abstract and general to deep-level analysis and repairing of detected deficiencies.

*Multi-format modelling.* To enhance the communicability and collaboration within a development and support team, the same model fragment must be provided in a number of formats in a coherent and consistent way.

*Reasoning in spite of errors.* Many modelling approaches require that the model must be complete before the analysis can start. We claim that is should be made possible to start analysis and model reasoning much earlier and incrementally.

*Constructive feedback from tools.* Instead of just pointing out problems, future tools should assist in resolving them.

*Support for evolution.* In general, applications keep evolving due to changes in the application domain, to changes of technology, changes in information systems purposes etc. A more constructive approach should also help managing the evolution of models.

*Support for reuse.* Problems in the application domain considered are more likely to be similar than solutions. Models reuse should therefore be even more promising than code reuse.

*Measurability of modelling progress.* To be more convincing, the benefits of using information

models should be measurable as well as their deficiencies.

This list of theories, solutions and methodological approaches is not exhaustive. It demonstrates, however, that modelling in the large and modern information systems modelling require specific approaches beyond integration of architectures into the analysis, design and development process.

## 2.5 The Research Issue

Let us reconsider the notions presented in Section 1. Table 1 compares essential properties of models. Missing model elements are denoted by n(ot).g(iven).

We observe that dependability is often either implicit or not considered in the model notion. Implicitness is mainly based on the orientation to normal models. The model matrix and especially the deep model are considered to be agreed before developing the model.

The origin is too wide in most cases. Models are not oriented towards representing some reality or the world. They are typically based on some kind of agreement made within a community of practice and according to some context, i. e. they reflect some domain-situation model[11] or more generally some mental model[12]. They might represent a perception model of some members of the community practice. They say what the phenomena in the given domain are like.

Table 1 directs to a conclusion that the function is mainly oriented towards description and partially prescription for systems development. The notion of the conceptual model has, however, mainly considered in system construction scenarios.

Concepts are often hidden behind the curtain of conceptual models. A conceptual model does not reflect the reality. Instead it reflects the mental understanding within its utilisation scenario.
These observations show now directly some open issues that should be solved within a theory and practice of conceptual modelling. Let us state some of them.

***Research question 1.***
What are the *origins* for conceptual models? Are these mainly domain-situation and perception models from one side and systems on the other side?

***Research question 2.***
How tightly are conceptual models bound to their *modelling matrix* and especially their *deep model*? To what extent are conceptual models normal models that are intentionally combined with their deep models?

***Research question 3.***
Which functions have conceptual models in which utilisation scenarios? Which properties must be satisfied by conceptual models in these scenarios? Which purposes and goals can be derived?

***Research question 4.***
What is the role of the *community of practice* in conceptual modelling? Which kind of model supports which community in which context?

***Research question 5.***
Conceptual modelling is less automated and more human dependent than any other development, analysis, and design process for information systems. It is a highly creative process. Is there any formalisation and foundation for this process?

[11] We restrict consideration to our field and thus to domain-oriented models. These models describe the application domain and more specifically the understanding, observation, and perception of an application domain that is accepted within a community of practice. In general, a situation model is a mental representation of a described or experienced situation in a real or imaginary world (see Radvansky and Zacks 1997).

[12] Mental models are out-of-scope in this paper. Those consist of an evolving model suite with small-scale and parsimonious models carried in human head (see Forrester 1971; Johnson-Laird 1983). They support various kinds of observation, information acquisition and filtering, reasoning, storage and information (de)coding, and communication. They are dependent on the observations, imaginations, and comprehension a human has made. Unlike conceptual models, mental models must neither be accurate, nor complete, and not consistent.

*Table 1: Orientation of notions of conceptual models according model properties*

| version | adequate | dependable | origin | function | scenario | concepts |
|---|---|---|---|---|---|---|
| reality, world | reflection, truncation | formal, reflection | world | describe | communication, understanding | n.g. |
| community | abstraction, mapping | semantic invariance | software application | describe | construction | n.g. |
| conceptual database | mapping, homomorphy | n.g. | data, operations | describe | construction, documentation | reality concepts |
| system & instance | mapping, abstraction | n.g. | system, objects | n.g. | construction | n.g. |
| system representation | reflection, | qualitative assumptions | system | describe | representation | system concepts |
| represen-tional | mapping | n.g. | relation-ships | represent | visualisation | impact factors |
| enterprise | mapping, abstraction | faithful | domain | purpose-determined | understanding | concept space |
| result of activity | mapping, | n.g. | system knowledge | describe | acquisition, elicitation | domain knowledge |
| purpose-oriented | abstraction purposeful | viable, fit | any | elicitate | n.g. | n.g. |
| documen-tation | summary, abstraction | n.g. | data system | represent, survey | strategy development | n.g. |
| semiotics | syntax abstraction | semantics, pragmatics | software application | describe | representation | n.g. |
| document understand | mapping | closeness | application | understand by users | design | n.g. |
| conceptualise | formal representation | semantics | any | describe | representation | concept(ion) space |
| ad-hoc | selective mapping | n.g. | domain | consider problem | solving | n.g. |

***Research question 6.***
Since models must not be conceptual models (see models in Thalheim and Nissen 2015), we might ask whether there exists a set of characteristics or criteria that separate a conceptual model from a model that is not conceptual. What is the concept space that can be used for an enhancement of a model by concepts or conceptions?

## 3 The Nature of Models

### 3.1 The Notion of a (Conceptual) Model

The model is an utterance and also an imagination. As already stated above (see also Thalheim and Nissen 2015), a model is a *well-formed*, *adequate*, and *dependable* instrument that represents *origins* and *functions* in some utilisation *scenario*. A model is a representation of some origins and may consist of many expressions such as sentences. *Adequacy* is based on satisfaction of the purpose or function or goal, analogy to the origins it represents and the focus under which the model is used. *Dependability* is based on a *justification* for its usage as a model and on a *quality certificate*. Models can be evaluated by one of the evaluation frameworks. A model is *functional* if methods for its development and for its deployment are given. A model is *effective* if it can be deployed according to its portfolio, i. e. according to the tasks assigned to the model. Deployment is often using some deployment macro-model, e.g. for explanation, exploration, construction, documentation, description and prescription.

Models *function* as *instruments* or tools. Typically, instruments come in a variety of forms and fulfil many different functions. Instruments are partially independent or autonomous of the thing they operate on. Models are however special instruments. They are used with a specific intention within a utilisation scenario. The quality of a model becomes apparent in the context of this scenario.

Model development is often targeted on normal models and implicitly accepts the deep model. A model is developed for some modelling scenarios and thus biased by its modelling matrix. The deep model and the matrix thus 'infect' the normal model.

Within the scope of this paper, we concentrate on representation models as proxies. So, a model of a collection of origins, within some context, for some utilisation scenario and corresponding functions within these scenarios, and for a community of practice is

- a relatively enduring,
- accessible
- but limited
- internal and at the same time external
- representation of the collection of origins.

The model becomes *conceptual* by incorporation of concepts and conceptions commonly accepted, of ideas provided by members from the community of practice, or of general well-understood language-like semiotic components. One main utilisation scenario for a conceptual database model is system construction[13]. In this case, the conceptual model thus becomes predictively accurate for the system envisioned and technologically fruitful. The model is an utterance and also an imagination. Other scenarios for conceptual models are: system modernisation, explanation, exploration, communication, negotiation, problem solving, supplantation, documentation, and even theory development.

Conceptual models must not be limited to the representation of static aspects of systems. They can also be used for the representation of dynamic aspects such as business stories, business processes, and system behaviour. The carrier of representation is often some language. In this case, a conceptual model can be considered to be an utterance with a collection of speech acts. The model itself can be then build on well-formedness rules for its syntax, semantics, and pragmatics, or more general of semiotics and semiology. According to J. Searle (see Searle 1969), a speech act consists of uttering elements, referring and predicating, requesting activities, and causing an

[13] Notice however that the first introduction of conceptual data models has been oriented on a documentation scenario.

effect. Whether at all and which language is going to be used is a matter of controversy as well.

## 3.2 Facets of a Conceptual Model

***1. The conceptual model is a result of a perception and negotiation process.***
The conceptual model represents mental models, especially domain-situation models or a number of perception models. Domain-situation models represent a settled perception within a context, especially an application. Perception models might differ from the domain-situation model. They are personal perceptions and judgements of a member of the community of practice. Maturity of conceptual models is reached after the community of practice negotiated different viewpoints and has found an agreement.

***2. The conceptual model represents its collection of origins.***
Considerations, about what to model and what to model not, are expressed via the adequacy criteria, especially for analogy to its origins, for focusing on specifics of the origins, and also on well-formedness of the model. The conceptual model does not represent the real world or a problem domain. It is already based on perception models of users about this problem domain or on domain-situation models of a user community on this problem domain.

***3. The conceptual model is an instrument.***
The conceptual model is used in some utilisation scenario by its users. So it functions in this utilisation scenario. It should describe, in a more abstract way compared to the origins, how the user conceives it and thus does not target on describing the origins.

***4. The deep model underpins the conceptual model.***
The deep model consists of all elements that are taken for granted, are considered to be fixed, and are common within the context for the community of practice. Elements of this model are symbolic generalizations as formal or readily formalisable components or laws or law schemata, beliefs in particular heuristic and ontological models or analogies supplying the group with preferred or permissible analogies and metaphors, and values shared by the community of practice as an integral part and supporting the choice between incompatible ways of practising their discipline. There is no need to redevelop this model. So, the normal model only display those elements that are additionally introduced for the model.

***5. The conceptual modelling matrix.***
The modelling matrix combines the deep model with typical utilisation scenarios, that are accepted by a community of practice in a given context. It specifies a guiding question as a principal concern or scientific interest, that motivates the development of a theory, and techniques as the methods a developer uses to persuade the members of the community of practice to his point of view. Although often not explicitly stated, the model matrix consists of a number of components: the objectives, inputs (or experimental factors), outputs (or responses), content requests, grounding, basis, and simplifications. The matrix sets a definitional frame for the normal model. It might support modelling by model stereotypes. The agenda of the modelling method is derived from the matrix. The matrix determines also a specific treatment of adequacy and dependability for a model.

***6. The performance and quality criteria.***
The model is a persistent and justified artefact that satisfy a number of conditions according to its function such as empirical corroboration according to modelling objectives, by rational coherence and conformity explicitly stated through conformity formulas or statements, by falsifiability, and by stability and plasticity within a collection of origins. The quality characteristics bound the model to be valid, credible, feasible, parsimonious, useful, and at the same time as simple as possible and as complex as necessary.

***7. The model is the main ingredient of a modelling method.***
Sciences and technologies have developed their specific deployment of models within their investigation, analysis, development, design etc. processes. The deep model and the matrix are often

agreed. The central element of all modelling methods is the model, that is used as an instrument in scenarios which have been stereotyped for the given modelling method. The modelling method typically also includes the design of a representation model (or a number of such). The representation model of the (conceptual) model may be based on approaches such as diagramming and visualisation. It uses a set of predefined signs: icons, symbols, or indexes in the sense of Peirce.

### 3.3 Sources for Conceptual Models: Domain-Situation and Perception Models

The domain-situation model is build by a community of practice on a semantical level. It refers to the world-as-described-and-conceived-by-the-deep-model. It thus forms the deep understanding behind the conceptual model. This deep internal structure of the conceptualisation is commonly shared in the community, abstracts from accidental origins, uses a partial interpretation, exhibits (structural) hidden similarities of all origins under consideration, and presents the common understanding in the community. It gives thus a literal meaning to the domain. The context for the conceptual model is typically governed by domain-situation models. The domain-situation model is thus one source for the conceptual model.

A domain-situation model might or might not exist. It shapes, however, what is seen in an application domain and how to reason about what is seen. It represents some common negotiated understanding in the application domain. It may represent the application domain as it is, or the application domain as it makes sense to be characterised, categorised or classified in one way rather than another, given certain interests and aptitudes or more generally given certain background.

The second source for conceptual models is a collection of perception models, that are provided and acknowledged by members of this community of practice. A perception model is one kind of epistemological mental model with its verbal, visual and other information compiled on the basis of cognitive schemata. It organises, identifies, and interprets observations made by the member. It does not need to know the deep facts or essential properties of the origins in order to succeed in communicating about them or to reason. The perception model typically follows the situation that it represents. It is, however, often undetermined and thus may also partially contradictory. So it parallels and imitates parts of the reality ('Gestalt' notion of the model). They provide a partial understanding, refer to some aspect, may use competing sub-models about the same stuff, and may set alternatives on meaning. It is build by intuitive, discursive and evidence-backed perception, by imagination, and by comprehension. It is shaped by learning, memorisation, expectation, and attention. Perception models serve as an add-on beyond domain-situation models.

These two sources for conceptual models depend on the community of practice. So, different communities might use different kinds of verbal and non-verbal representation. Although they provide a literal meaning to the conceptual model, they must not be explicitly stated within the conceptual model. They serve as the origin for the conceptual model and thus might not be explicitly incorporated into the conceptual model. The conceptual model may have its deep background, i. e. its basis and especially its grounding.

Both origins are not complete. Typically the scope of both models is not explicit. There are unknown assumptions applied for description, unknown restrictions of the model, undocumented preferences and background of the community of practice, and unknown limitations of the modelling language. Classically, we observe for members of a community of practice that:

- they base their design decisions on a "partial reality", i. e. on a number of observed properties within a part of the application,
- they develop their models within a certain context,
- they reuse their experience gained in former projects and solutions known for their reference models, and

- they use a number of theories with a certain exactness and rigidity.

The conceptual model to be developed is deeply influenced by these four hidden factors.

## 4 Conceptualisation of Models

The domain-situation model and also partially the perception model are using concepts commonly. Conceptual models reuse such concepts from these origins and thus inherit semantics and pragmatics from these models. Further, conceptualisation may also be implicit and may use some kind of lexical semantics of these models, e. g. word semantics, within a commonly agreed name space.

### 4.1 Concepts and Conceptions

Various notions of concept has been introduced, for instance, by J. Akoka, P. Chen, H. Kangassalo, R. Kauppi, A. Paivio, and R. Wille (see Chen et al. 1998; Ganter and Wille 1998; Kangassalo and Palomäki 2015; Kauppi 1967; Paivio 1986). Artificial intelligence and mathematical logics use concept frames. Ontologies combine lexicology and lexicography. Concepts are used in daily life as a communication vehicle and as a result of perception, reasoning, and comprehension. Concept definition can be given in a narrative informal form, in a formal way, by reference to some other definitions etc. Some version may be preferred over others, may be time-dependent, may have a level of rigidity, is typically usage-dependent, has levels of validity, and can only be used within certain restrictions. We also may use a large variety of semantics (see Schewe and Thalheim 2008), e.g., lexical or ontological, logical, or reflective.

We distinguish two different meanings of the word 'concept' (see White 1994):

1. Concepts are general categories and thing of interest that are used for classification. Concepts thus have fuzzy boundaries. Additionally, classification depends on the context and deployment.
2. Concepts are all the knowledge that the person has, and associates with, the concept's name. They are reasonable complete in terms of the business.

*Conceptions* (see White 1994) are systems of explanation. They are thus more difficult to describe.

The typical definition frame we observed is based on definition items. These items can also be classified by the kind of definition. Concepts may have different descriptions simultaneously. A competing description may represent the same concept differently depending on the context (e. g. time, space), validity, usage, and preferences of members of the community of practice. A concept may have elements that are necessary or sufficient, that may be of certain rigidity, importance, relevance, typicality, or fuzziness. Based on the generalisations of the approach, that has been proposed by G.L. Murphy (see Murphy 2001; Thalheim 2007), concepts are defined in a more sophisticated form as a tree-structured structural expression.

```
SpecOrderedTree(StructuralTreeExpression
        (DefinitionItem,
          Modality(Sufficiency, Necessity),
          Fuzziness, Importance, Rigidity,
          Relevance,
          GraduationWithinExpression,
          Category))) .
```

Concept may be regarded as the descriptive and epistemic core units of perception and domain-situation models. These origins govern the way how a concept can be understood, defined, and used in a conceptual model. The conceptual model inherits thus concepts and their structuring within a concept space, i. e. conceptions.

### 4.2 Conceptualise

Conceptualisation and semantification are orthogonal concerns in modelling. *Conceptual modelling* is based on concepts that are used for classification of things. Concepts have fuzzy boundaries. Additionally, classification depends on the context

and deployment. Conceptual[14] modelling uses *conceptions*, which are systems of explanation.

*Semantification* (see Duží et al. 2009) improves the comprehensibility of models and explicit reasoning on elements used in models. It is based on name spaces or ontologies which are commonly accepted in the application domain. Conceptual models are models enhanced by concepts and integrated in a space of conceptions.

Conceptualisation injects concepts or conceptions into models. These enriched models reflect those concepts from commonly accepted concept space. The concept space consists of a system of conceptions (concepts, theoretical statements (axioms, laws, theorems, definitions), models, theories, and tools). A concept space may also include procedures, conceptual (knowledge) tools, and associated norms resp. rules. Is is based on paradigms which are corroborated.

### 4.3 Dependability of Conceptual Models

Models must be dependable, i. e. justified from one side and and qualitatively certified from the other side. Justification can be based on the domain-situation and perception models and the relation of the conceptual models to these models. If, however, such models are not available or of low quality, justification will become an issue. Quality certification is an issue of pragmatism and of added value of the conceptual model. So, we target on a high quality conceptualisation.

Conceptualisation may be based on the seven principles of Universal Design (see Patil et al. 2003). Typical mandatory principles are usefulness, flexibility, simplicity, realisability, and rationality. Optional conceptualisation principles are perceptibility, error-proneness, and parsimony.

The *principle of conceptualisation* is considered to be one - if not the main - of the seven fundamental principles for conceptual modelling (see Griethuysen 2009). The other six principles are: Helsinki, Universe of discourse, searchlight, 100%, onion, and three level architecture principles. They can be questioned further. These principles can be enhanced by the principles of understanding, of abstraction, of definition, of refinement, evaluation, and of construction (see Thalheim 2010). Conceptualisation can be considered to be completed if: A conceptual schema should only include conceptually relevant aspects, both static and dynamic, of the universe of discourse, thus excluding all aspects of (external or internal) data representation, physical data organization and access, as well as all aspects of particular external user representation such as message formats, data structures, etc.
Based on Section 3.3, the principle of conceptualisation can be stated as follows:
*A conceptual model should only include conceptually relevant aspects of the domain-situation and perception models. It does not consider neither aspects of realisation nor of representation. It includes, however, different viewpoints of business users and concepts from the common concept space.*

## 5 Conclusion: Towards a Notational Frame for Conceptual Models

Conceptual modelling is not yet a science or culture. It is rather a craft or even an art. It can be learned similar to craft learning. It is however based on understanding and abstraction throughout the perception and domain-situation models, i. e. of mental models in general. Perception is dependent on deep models and thus incomplete, revisable, time-restricted, activity-driven, and context-dependent.

### 5.1 Slim, Light, and Concise Versions for Conceptual Models

Conceptual models are widely used in system construction scenarios. They function as description of the phenomena of interest within the context for its community of practice. So, conceptual models are normal models with rather specific modelling matrices and deep models. A slim notion of a

[14] Conceptual modelling is performed by a modeller that directs the process based on his/her experience, education, understanding, intention and attitude. Conceptual models are using/incorporating/integrating concepts (see White 1994) Conceptional modelling aims at development of concepts.

conceptual model should only reflect such normal models and refer to a specific modelling matrix. A light version needs to refer to some elements of the basis and to some context. A concise version must explicitly represent all the hidden details of a model, especially its relationships to the concept space, to the perception of this space by members of the community of practice, and to the utilisation scenario.

### 5.2 A Proposal for a Light Version: Conceptual Model $\sqsupseteq$ Model $\bigoplus$ Concepts

Conceptual modelling is not yet a common method in science (see Robinson 2010). Systems can be build without any conceptual model. It seems that there is no need for a formal conceptual modelling process. It seems to be too restrictive to require a full conceptual model. Performance and quality criteria are not commonly agreed. The science of conceptual modelling is still missing.

The main bottleneck is however the missing notion of a conceptual model. The conceptual model is a specific model and is based on conceptualisation. It might be language-bound. It is probably the most important aspect of system construction in computer science and computer engineering. It is however the most difficult and least understood. Minimal justification characteristics of models are classical viability, i. e. corroboration, validity, credibility, rational coherent and conform, falsifiable, stability against origin collection change. Minimal quality characteristics of models are the one for quality in use (e.g. usability, aptness for the function and purpose, value for the utilisation scenario, feasibility). Minimal performance characteristics are timely, elegant and feasible usage within the given context for their community of practice according to their utilisation scenario and their competencies or more general their profiles.

So, we might conclude for a *light version*: *A conceptual model is a well-formed, adequate and dependable instrument that functions within its specific utilisation scenario, that represents origins, and that is enhanced by concepts from a concept(ion) space.*

Therefore, the incorporation of concepts and the conceptions is one main difference to the model.

### 5.3 Lacunas of Conceptual Modelling

Since conceptual modelling is still more an art than a science and a culture of conceptual modelling is still beyond the horizons, we need

- an understanding of the area of conceptual modelling;
- a theory, techniques, and engineering of conceptualisation;
- an integrated multi-view approach for the needs and the capabilities of the members of the community of practice;
- a refinable definition of the conceptual model with all three versions, i. e. a simplified version, a fully fledged version, and an assessable version;
- a working approach with intentional and thus latent matrices and deep models for daily practice; and
- an understanding of language use in conceptual modelling.

These lacunas do not limit usability, usefulness, and utility of conceptual models. Conceptual database models improve, from one side system, comprehension. They allow to indicate associations among system elements, reduce the effect of bad implementation, provide abstraction mechanisms, support prediction of system behaviour, provide an elegant and adequate overview of the system at various levels of abstraction, support the construction of different user views, and cross-reference multiple viewpoints. From the other side, they reduce developers, maintainers and programmers overhead. They support a simple and free navigation through components of the database system, provide an easy deduction of various viewpoints, that represent the needs of business users, support concentration and focusing in evolution and maintenance phases, display the decisions made during development, indicate opportunities for further development and system maintenance, reduce the effort by reuse of design

and development decisions that have already been made, and use a comfortable and effective visualisation. So, conceptual models are not restricted to construction scenarios or to database modelling.

We realise, that the development and the acceptance of a notion of conceptual model follows the 13 commandments stated (see Bowen and Hinchey 2009):

1. Thou shalt choose an appropriate notation.
2. Thou shalt formalise but not over-formalise.
3. Thou shalt estimate costs.
4. Thou shalt have a formal methods guru on call.
5. Thou shalt not abandon thy traditional development methods.
6. Thou shalt document sufficiently.
7. Thou shalt not compromise thy quality standards.
8. Thou shalt not be dogmatic.
9. Thou shalt test, test, and test again.
10. Thou shalt reuse.
11. Thou shall meet intentions of all members of the community of practice
12. Thou shall provide a usable notation, i. e. for verification, validation, explanation, elaboration, and evolution.
13. Thou shall be robust against misinterpretation, errors, etc.

## References

Ambler S., Sadalage P. (2006) Refactoring databases - Evolutionary database design. Addison-Wesley

Batini C., Ceri S., Navathe S. (1992) Conceptual database design (an entity-relationship approach). Benjamin/Cummings, Redwood City

Bjeković M. (2017) Pragmatics of Enterprise Modelling Languages: A Framework for Understanding and Explaining. PhD thesis, Radboud University Nijmegen

Boman M., Bubenko Jr. J., Johannesson P.and Wangler B. (1997) Conceptual modelling. Prentice Hall, London

Bowen J. P., Hinchey M. G. (2009) Ten Commandments Ten Years On: Lessons for ASM, B, Z and VSR-net. In: Rigorous Methods for Software Construction and Analysis. Lecture Notes in Computer Science Vol. 5115. Springer, pp. 219–233

Business dictionary (2017) Conceptual model. http://www.businessdictionary.com/definition/-conceptual-model.html. Last Access: Assessed Nov. 21, 2011

Chen P., Akoka J., Kangassalo H., Thalheim B. (eds.) Conceptual Modeling: Current Issues and Future Directions. LNCS Vol. 1565. Springer

Duží M., Heimburger A., Tokuda T., Vojtas P., Yoshida N. (2009) Multi-Agent Knowledge Modelling. In: Information Modelling and Knowledge Bases XX. Frontiers in Artificial Intelligence and Applications, 190. IOS Press, pp. 411–428

Elmasri R., Navathe S. (2000) Fundamentals of database systems. Addison-Wesley, Reading

Embley D., Thalheim B. (2011) Preface. In: The Handbook of Conceptual Modeling: Its Usage and Its Challenges. Springer, Berlin, pp. v–ix

ER community (2017) Homepage. http://conceptualmodeling.org. Last Access: Assessed Oct. 29, 2017

Forrester J. (1971) Collected papers of J.W. Forrester In: Wright-Allen Press, Cambridge chap. Counterintuitive behaviour of social systems, pp. 211–244

Ganter B., Wille R. (1998) Formal concept analysis - Mathematical foundations. Springer, Berlin

Griethuysen J. J. V. (2009) The Orange Report ISO TR9007 (1982 - 1987) Grandparent of the Business Rules Approach and SBVR Part 2 - The Seven Very Fundamental Principles. In: Business Rules Journal 10(8) http://www.brcommunity.com/a2009/b495.html

Halpin T. A. (2009) Object-Role Modeling. In: Encyclopedia of Database Systems. Springer US, pp. 1941–1946

InfoAdvisors (2017) What are Conceptual, Logical and Physical Data Models? http://www.datamodel.com/index.php/articles/what-are-conceptual-logical-and-physical-data-models/. Last Access: Assessed Nov. 21, 2011

Johnson-Laird P. (1983) Mental models: Towards a cognitive science of language, inference, and consciousness. Cambridge University Press, London

Johnson J., Henderson A. (2013) Conceptual modelling in a nutshell. http://boxesandarrows.com. Last Access: Assessed Jan. 29, 2013

Kangassalo H., Palomäki J. (2015) Defintional conceptual schema - The core for thinking, learning, and communication. Keynote given at 25th EJC Conference, Maribor, Slovenia

Karagiannis D., Mayr H. C., Mylopoulos J. (eds.) Domain-Specific Conceptual Modeling, Concepts, Methods and Tools. Springer

Kauppi R. (1967) Einführung in die Theorie der Begriffssysteme. In: Acta Universitatis Tamperensis, Ser. A, Vol. 15, Tampereen yliopisto, Tampere

Mayr H. C., Michael J., Ranasinghe S., Shekhovtsov V. A., Steinberger C. (2017) Model Centered Architecture. In: Conceptual Modeling Perspectives. Springer, pp. 85–104

Murphy G. L. (2001) The big book of concepts. MIT Press

Mylopoulos J. (1992) Conceptual modeling and Telos. In: Conceptual modeling, databases, and CASE: An integrated view of information systems development. LNCS 10509. Wiley, Chichester, pp. 49–68

Olivé A. (2007) Conceptual modeling of information systems. Springer, Berlin

Paivio A. (1986) Mental representations: A dual coding approach. Oxford University Press, Oxford

Pastor O. (2016) Conceptual Modeling of Life: Beyond the Homo Sapiens; Keynote at ER 2016. http://er2016.cs.titech.ac.jp/assets/slides/ER2016-keynote2-slides.pdf. Last Access: Assessed Nov. 15, 2016

Patil B., Maetzel K., Neuhold E. J. (2003) Design of International Textual Languages: A Universal Design Framework. In: IWIPS. Product & Systems Internationalisation, Inc., pp. 41–56

Radvansky G., Zacks R. (1997) Cognitive models of memory In: Convey M. (ed.) MIT Press, Cambridge chap. The retrieval of situation-specific information, pp. 173–213

Robinson S. (2010) Conceptual modelling: Who needs it? In: SCS M & S Magazine (2), pp. 1–7

Schewe K.-D., Thalheim B. (2008) Semantics in Data and Knowledge Bases. In: SDKB 2008. LNCS 4925. Springer, Berlin, pp. 1–25

Searle J. (1969) Speech Acts: An essay in the philosophy of language. Cambridge University Press, Cambridge, England

Thalheim B. (2000) Entity-relationship modeling – Foundations of database technology. Springer, Berlin

Thalheim B. (2007) The Conceptual Framework To User-Oriented Content Management. In: Information Modelling and Knowledge Bases. Frontiers in Artificial Intelligence and Applications Vol. XVIII. IOS Press

Thalheim B. (2010) Towards a Theory of Conceptual Modelling. In: Journal of Universal Computer Science 16(20) http://www.jucs.org/jucs_16_20/towards_a_theory_of+, pp. 3102–3137

Thalheim B. (2014) The Conceptual Model ≡ An Adequate and Dependable Artifact Enhanced by Concepts. In: Information Modelling and Knowledge Bases. Frontiers in Artificial Intelligence and Applications, 260 Vol. XXV. IOS Press, pp. 241–254

Thalheim B. (2017) General and Specific Model Notions. In: Proc. ADBIS'17. LNCS 10509. Springer, pp. 13–27

Thalheim B., Nissen I. (eds.) Wissenschaft und Kunst der Modellierung: Modelle, Modellieren, Modellierung. De Gruyter, Boston

van Lamsweerde A. (2000) Formal specification: a roadmap. In: ICSE - Future of SE Track, pp. 147–159

van Lamsweerde A. (2008) Requirements engineering: from craft to discipline. In: SIGSOFT FSE. ACM, pp. 238–249

White R. (1994) Commentary: Conceptual and conceptional change. In: Learning and instruction 4, pp. 117–121

Whorf B. (1980) Lost generation theories of mind, language, and religion. Popular Culture Association, University Microfilms International, Ann Arbor, Mich.

Wikiquote (2017) Conceptual model. https://en.wikiquote.org/wiki/Conceptual_model. Last Access: Assessed Nov. 21, 2017

WordNet dictionary (2017) Conceptual model. http://www.dictionary.com/browse/conceptual-model. Last Access: Assessed Nov. 21, 2011

Zeigler B. P., Kim T. G., Praehofer H. (2000) Theory of Modeling and Simulation. Elsevier Academic Press

# 20 Years After: What in Fact is a Model?

Roland Kaschek*,a

[a] Alpha-Academica

Abstract. *Heinrich C. Mayr, for many years has been a leading proponent of the importance of modelling issues for computing, the GI and the EMISA. On occasion of his 70th. birthday, this paper sheds a light on a number of key modelling issues.*

Keywords. Model • Theory of Modelling • Thesaurus • Reference Modes • Models as Media Content

## 1 Introduction

Time is a human invention for structuring narratives. It is therefore that we look back into time when we encounter anniversaries in narratives. In narratives concerning modern computing in Germany the name Dr. Dr. h. c. Heinrich C. Mayr figures highly. His 70th. birthday is thus an anniversary to respond to. In particular it means a lot to me personally. I contribute this paper to thank him for the opportunities he gave me.

Heinrich was one of the founding members of the EMISA. He is a long time EMISA-activist and was for many years a leader of the GI. His impact on modern computing in Germany is not second to that of many other colleagues. I wish him very well for any of his future plans and in particular personal well-being.[1]

In 1999, I have addressed the question what a model actually is in this journal as a current catch phrase and turn to this subject again now. I comment on earlier work I started as one of Heinrich's post docs with the then University of Klagenfurt and that I have continued when I was with Massey University in New Zealand. The question, as to what a model actually is, is still being discussed (Sander 2011, Ungermann 2017). I have worked on modelling issues for many years and try to add some new stuff now to that earlier discussion.

How one defines the term model is a question of convenience. In my current view traditional definitions of the term model are not really convincing. I thus in this paper try to contribute to a common ground based on which a more suitable definition might emerge and gain general acceptance. My text (Kaschek 1999) is not available on the EMISA Web site. I thus include a brief summary of its key points in this paper. Right now, I can not go into a full review of the scientific discussion on modelling since 1999. Therefore, I limit myself in this paper to adding some aspects of it that I have introduced since then and to extending or modifying points where that seems appropriate.

## 2 Basic modelling narratives

From old age, models have been created and used to impact the quality of life. Let it be that an improvement of it was intended, its deterioration was to be prevented from happening or a current state of affairs was to be understood. Models are the result as well as the origin of narratives of a practice related to these livelihoods. The meta practice of handling models may increase the capacity of shaping the related practice. All pursuits start with the most obvious. Thus, modelling starts with using the human body and in particular its limbs in a deictic way. What would be found

* Corresponding author.
E-mail. rolandkaschek@gmail.com

[1] Note that this paper strictly spoken is not a scientific paper, as it contains several views that I do not attempt to prove or even justify.

near by, such as sticks, stones, or similar, probably would have been used too. With sign-systems external to the human body, in particular script, coming into use, models increasingly become a common good, a cultural asset, and the mentioned meta practice becomes a shared one. Obviously that does not mean that all the participants in the meta practice have the same interests in practice or would work on the same task. Further recent remarks concerning origin and history of modelling can be found in (Hesse and Mayr 2008).

Computer-use has become ubiquitous. Therefore software production and use has become a mayor branch of the economy of the developed countries. Computing, among others, contributed to that by providing methods and tools for software development. In particular, it has provided abstractions, that are able to specify the business processes that exist, how they are related to each other, how they are being executed, what data they refer to and how that data is being structured and processed. Together with an increasing capability to provide new or improved products or services, which has contributed to computers penetrating the economy. Many of these abstractions required for this process have been conceptualized as model. Models and modelling, to some extent, can thus be understood as enablers of the modern economy.

The interest of computing in modelling overwhelmingly seems to result from its task, i. e., to schedule men power, financial, organizational and technical means as required for improving or even enabling the business models, services or products of certain organizations, such as corporations or authorities. The shared meta practice chiefly rests on a particular part of the model, the so-called thesaurus, i. e., the list of the definitions of the shared key concepts including the instructions of how to refer to the items potentially corresponding to these concepts. Despite its paramount importance for modelling, to my knowledge, the thesaurus in many modelling approaches and papers on modelling is undervalued.[2] To deal with that, one ought to work on how to produce a usable thesaurus, how to define, measure and improve the quality of a thesaurus.

Attraction and power of the early sense-giving or explanatory narratives probably result from their poetry and mysticism, as well as the authority of those who were in control of those narratives. With the development of technology, the emergence of the division of labour and in particular institutions of education and learning, narratives emerge whose utility transcends the organization of human communities and opens up the space of nature control. However, it nearly goes without saying, that control of nature is an illusion most of the time and in fact is a consequence of decisions to ignore certain aspects of it. Practical control over nature is or extends also, at least potentially, into power over people. The sciences and modelling thus may be objective. They are not neutral, however. An alternative to nature control is available in form of policies to nature appropriation that aim at men's purposefully integration into nature. An assessment of the consequences of using given models, in general only rarely takes place. Furthermore, it would probably be limited to private enterprises and ignore public spaces and communities. Any detrimental consequences of model use, that way, are largely blacked out from the planning that takes place in private enterprises. As much of current modelling is embedded into software projects, that are run by private companies, it is quite likely that one, upon close examination, would find that narratives are emerging or even have already achieved dominance, that consider the interests of users only or mainly from the perspective of companies and thus might develop potential to harm human communities.

The two most obvious idealizing characterizations of an item refer to what it is in itself and to what it is for some individual. What a model is in itself, leads to the question of what its elementary parts are and how these are combined into a whole. What a model is for some individual, leads to the question, which operations the related individual applies to the model. Stachowiak (Stachowiak 1973; Stachowiak 1983; Stachowiak 1992), who

[2] As an example of a modelling paper that entirely ignores the thesaurus is (Hesse and Mayr 2008).

I have used before (Kaschek 1999), with regard to these questions, says that models are sets of predicates and that model users use models to, in a controlled way, refer to something different from the model, i. e., a model original. That reference must be controlled, since it is supposed to enable an information transfer between model and original (and vice versa). This information transfer is potentially beneficial to the model user who even might change their practice because of the model's impact on that practice.

Model use is about an information deficit with regard to some item, a so-called model original: someone wishes to achieve a certain goal with regard to the original. They, however, turn out to be ignorant of some information (subjectively) required for achieving that goal. The key idea to solve this problem is to obtain the required information with regard to the model and then transfer it to the original.[3] For that to be a justifiable option, the model needs to share some of the original's characteristics. To enable the model user to process the model in a way, that provides them with the information needed, the model needs to have appropriate characteristics. Some of these won't be shared with the original. This basic narrative can serve as a justification for Stachowiak's decision to identify three key features of any model, namely the **mapping feature**, the **reduction feature** and the **pragmatic feature**. According to the mapping feature to each model there is an original. According to the reduction feature a model will share some of the original's characteristics, but might miss out on some others. The pragmatic feature finally means that the association of an original to one of its models is not a once and for all established one. Rather, a voluntary decision of the model user introduces or drops that association. Consequently, the model-original relationship is a many-to-many relationship. The originals to a model, sort of, are similar to a quantum cloud of items that may be referred to from the very model. The mentioned information transfer, however, is not a one-way-street. Information about the original feeds the process of model creation. The two mappings applied to accomplish the required information transfers are not necessarily inverse to each other.

The reduction feature requires particular attention since it is in general complemented with an extension feature (already observed by Stachowiak). The feature reduction is due to keeping the model small and at the same time relevant with regard to the pursued goal. However, the model is supposed to undergo certain manipulation or operations that have to be carried out by certain individuals under specified conditions. The model thus in general needs to have characteristics it does not share with an original. As far as I see that crucial point was not observed by Stachowiak. It also is not discussed in (Hesse and Mayr 2008). The extension feature (or, as Stachowiak has referred to it, the abundance of model characteristics over original characteristics) thus is not a defect or accidental feature of models, it is rather essential for models to be able to serve the model users in the intended way.

In retrospection, it occurs to me that Stachowiak can be critiqued for insufficiently dealing with the quality of the model-original reference and the various kinds of such references that occur in modelling. In my view, the quality of model-original references is essentially determined by the thesaurus. So, I think, Stachowiak's theory should be extended by discussions of the thesaurus and what I call reference modes.

After a model user has established a model-original-reference, the information transfer from model to original is about utilizing the model. Modelling thus belongs to what is known as analysis-synthesis cycle. Models usually result from analysis and extend the epistemic, manipulation and control powers concerning the original. The model then may be used to state hypothesis with regard to one of its originals. Changes or invariances with regard to an original may become predictable and an original may become controllable. In any case, the often even huge differences

[3] In the light of this context I hold that information can be defined as any item, that can be fed into a decision procedure, which is supposed to achieve a given goal. See the discussion in (Sloman 2016).

between model and original may result in the mentioned information transfer even being challenged. In particular, because empiric evidence in favour of that transfer may be the only or key evidence justifying that transfer and moreover the model in several ways may even be wrong.[4] However, as long as the model is sufficiently useful one would overlook imperfections or defects of the mentioned kind as long as there is no viable alternative.

Gödel's incompleteness theorems (Boolos et al. 2010), in my view, are the deepest reason for the rise of modelling. These theorems say that (1) the consistency of a sufficiently expressive theory cannot be proven by means of that theory alone and that (2) in a sufficiently expressive consistent theory propositions may be stated, that cannot be proven within that theory. In view of the relative boundaries for establishing shared certainty based on rational narratives, a resorting to shared rules of knowledge discovery and decision making may provide an alternative to relapsing to poetry or mysticism of the basic narratives.

The Gödel theorems, with regard to the epistemic process, imply that its methodological base and its logical foundations have to be adapted continuously if an ever extended understanding and control of nature is intended. These theorems put the ones searching for knowledge about the world into the shoes of a tourist whose car occasionally needs to get fuelled up, in order to get to the desired point of interest.

In narrative based epistemic communities narration traditions may emerge with regard to sensegiving or explanatory narratives. These may even turn into epistemic obstacles, with regard to which modelism may turn out to be useful due to its anarchic unconcern due to its strong fixation at results.

[4] To briefly comment on the relationship between empiric and analytical knowledge, I state my view, that empirical knowledge, in one or another way, is ultimately processed into analytical knowledge. The latter, moreover, in fact may not be really analytical. It rather may be tied to the part of the universe in which it has emerged and might turn out to be utterly useless in distant parts of that very same universe.

## 3 Reference modes

A number of dimensions can be identified that a discourse on modelling could need to focus on. In this note I, more or less, focus on efficacy and function of a model. In earlier papers (Kaschek and Mayr 1996, 1998) Heinrich and I have discussed tool and method aspects of models, respectively. The dimensions I want to put forward are: **class**, the discourse in this dimension would focus on the various instances of the model, if any are allowed to exist; **context**, the discourse in this dimensions would focus on the methods that employ models; **definition**, the discourse in this dimension would focus on whether the model concept is defined explicitly, implicitly or not at all and if it is defined, what definition is given; **efficacy**, the discourse in this dimension would focus on how well references from a model to its originals can be carried out; **function**, the discourse in this dimension would focus on the kind of reference that is made from a model to its originals; **impact**, the discourse in this dimension would focus on the consequences model use probably has for human individuals and communities; **information**, the discourse in this dimension would focus on the kind, quality and amount of information to be transferred from a model to one of its originals; **object**, the discourse in this dimension would then focus on the ways to conceptualize a model original as an entity, operation, process or similar; **objectivity**, the discourse in this dimension would focus on either understanding a model as an aggregation of multi-perspective constructions or as an in-itself-existing entity; **ontology**, the discourse in this dimension would focus on whether to consider a model as a system of signs, a physical object, an idea or something else; **pragmatics**, the discourse in this dimension would focus on the quality of the model; **state**, the discourse in this dimension would then focus on the life cycle stage a model is in and the operations that therefore appropriately may be used on it and **tools**, the discourse in this dimension would focus on the tools used when dealing with models.

Heinrich used to consider modelling from the point of view of model creation. To some extent, that view might justify focusing on structural aspects of models rather than on the operations model users apply to them. He has structured models into concepts for either items or relationships between items. While that certainly was inspired by the huge popularity of the entity-relationship modelling (Chen 1976), the basic idea also can be applied to object oriented modelling and process modelling. With regard to model usage, however, one would probably want model-handling-operations to be considered as more important. In particular, considering any data structured according to the model would have to be included as it might have to be used for referring to an original. In that regard (Batini et al. 1992) seems to be pioneering work.

A peculiarity of the original-reference from a model can be illustrated nicely using the concept of Turing Machine. In computing, Turing machines are frequently considered as model of the concept of algorithm (Boolos et al. 2010). That, however, does not require a common understanding of algorithms prior to Turing's invention of what is now called Turing machine. Rather, it was essentially the power of Turing's idea of an ideal computing device that helped bring about a consensus regarding what the term algorithm should mean. There are examples for similar circumstances. For example, Yudkin (2012, p. 77) talks about how to find out how the human metabolism processes sugar. To answer that, one would start figuring out and set up an experiment that then will be carried out. The probands, or maybe even the experimentation rules, would then be used as the model. It, however, would not be obvious, what the original should be. In the case of a well-designed experiment, it would probably, as it was in case of the algorithm, constitute the concept of human sugar metabolism.

When, some years back, the concept of ontology was becoming mainstream, I found that this constitutive model-original reference was used there too, i. e., the original was being created by means of a model. Reference modes, that I have identified earlier (Kaschek 2005), are: the **descriptive mode**, the model describes, how the original looks like (photography); **prescriptive mode**, the model describes, how the original has to look like (norm); **constitutive mode**, the model defines the original (ontology); **idealizing mode**, the model specifies the original's ideal state (proceeding model); **prognostic mode**, the model describes a previous or future original state (climate model) and the **explanatory mode**, the model explains certain aspects of the original (Bohr's atom model, Mendeleev's periodic table of elements, . . .).

In my view, the general discussion back then was seriously limited. It essentially ignored the way in which models were being used to refer to originals. It is, however, exactly this reference mode that tells which model original differences (MOD) supposedly are considered essential and what to do about them. This, obviously is what modelling is all about. Rather than attempting to work out the key reference modes, some work (such as Hesse and Mayr 2008) tried to characterize the models that were to be used in those modes. In my view, however, it was clear already a long time ago, that such attempts are futile, as any model can be used any way the model users see fit.

In case of the descriptive reference mode, a non-tolerable MOD has to be dealt with by changing the model. In case of the prescriptive reference mode, the original has to be adapted, however. In case of the constitutive mode, the problem does not occur. In idealizing mode, one might even argue that MODs have to be tolerated because the original fails to be in ideal state and cannot, at acceptable cost, turned into that state. In case of the prognostic mode, it is key that any important MODs essentially disappear at the right point in time. In the explanatory mode, it suffices that any important MODs can be explained by the model presuppositions or can be ignored altogether without challenging the model's explanatory power.

## 4 Models as media content

In Stachowiak's view, models consist of predicates. I find this view suspicious, as any predicates do not include the perspective from which the original is viewed and ultimately justifies the ascription of the predicates to the original. I think, that an approach to modelling would be superior, that uses judgements to ascribe characteristics to originals as these inherently include the perspective under which the judging individual or community views, i. e., constitutes, the original.[5] Obviously, the way originals are constituted, ultimately carries over to model specification. As theoretical foundation of the theory of judgements, in the past I have used (Pfänder 1921), see also (Kaschek 2004).

Since some time I adhere to Nietzsche's dictum (Nietzsche 1887) that any perception involves a perspective, i. e., involves a peculiar and personal way of constituting conceptually a thing under scrutiny. For that , the distinction of the thing by itself from the thing for someone in my view cannot really be followed through. Rather, in my view, it is a modelling approach of actually rather limited explanatory powers. I think that Nietzsche is also right in asserting that the best that can be hoped for is a multi-perspective perception of scrutinized entities. Therefore, I feel that attempts to structurally characterize models are insufficient. They need to be complemented by some approach that ignores the model structure.

Models can be understood as to facilitate a conversation between the creators and the users of the model. The information transfer mappings mentioned before, can be understood as being implemented by that conversation. Since, in that sense, modelling intimately is connected to a media facilitated communication, at this time I prefer a definition of the model concept as media content. Of course I stick to Stachowiak's model features. Moreover, I add the reference mode and the much increased role of the thesaurus. Considering models as media content makes it superfluous to consider syntactical details when it comes to distinguishing models from non-models, since, what counts simply is, whether or not it a valid media content. It might even help to include into modelling the perspective of model user more effectively than this has been achieved so far. I would not necessarily claim, though, that in each case a given media content is used to refer to something different from it. However, in my view that is actually pretty often the case.

The idea to consider models as media content that facilitates a conversation between model creators and model users suggests to understand the concept of model quality as the consistency of that model that contributes to that conversation being helpful with regard to the model users reaching the goal, that triggered the model use in the first place. At this time I can only contribute most basic related ideas. If one takes a transactional view of communication, according to which communication essentially is an exchange of messages, then, to understand communication one has to understand sequences of messages. (Schulz von Thun 2017) provides a simple, yet powerful, meta model of messages. Schulz von Thun distinguishes the following message dimensions, **content**, **me**, **you** and **appeal**. He, moreover, considers each message as a compound of message aspects according to each of the mentioned dimension.

By the content dimension of a message, an interlocutor expresses what the message actually is about, while in the me-dimension that interlocutor, to some extent and in some way, discloses to their partner how they view themselves and by the you-dimension, expresses, how they perceive and relate to the partner. Finally, in the appeal-dimension, the interlocutor expresses what they want the partner to do in response to the message reception.

Some of the things to be considered with regard to the virtual conversation of model-user with model-creators are: the **kind of item** that may, should, shouldn't or must not be referred to with the aid of the model; the **modes**, in which these items may, should, shouldn't or must not be referred to; the **goals**, that may, should, shouldn't or must not be pursued by means the model; for each

[5] As is well-known, in database integration this view is actually commonly taken.

**goal**, what are the processes, operations and data that are required to achieve the goal; how to assess the model user's **readiness** to use the model and how to bring that readiness to the required level; and finally, how to identify the **expectations** the model user has with regard to the model and how these can be communicated to the model creator.

It would be interesting to pursue further research with regard to any improvements of the theory of model quality, that can be achieved by considering models as media content. This, however, would have to be the subject of further papers.

## References

Batini C., Ceri S., Navathe S. B. (1992) Conceptual Database Design: An Entity-relationship Approach. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA

Boolos G., Burgess J., Jeffrey R. (2010) Computability and logic. Cambridge University Press

Chen P. (1976) The Entity-Relationship model – toward a unified view of data. In: ACM Transactions on Database Systems 1 (1), pp. 9–36

Hesse W., Mayr H. C. (2008) Modellierung in der Softwaretechnik: eine Bestandsaufnahme. In: Informatik Spektrum 31 (5), p. 307

Kaschek R. (1999) Was sind eigentlich Modelle? In: EMISA Forum

Kaschek R. (2004) Konzeptuelle Modellierung. Habilitation, Universität Klagenfurt

Kaschek R. (2005) Modeling ontology use for information systems. In: Professional Knowledge Management. LNCS 3782. Springer Verlag, pp. 609–622

Kaschek R., Mayr H. C. (1996) A characterization of OOA tools. In: Proceedings of The 4th. International Symposium on Assessment of Software Tools. IEEE Computer Society Press, Los Alamitos, California, pp. 59–67

Kaschek R., Mayr H. C. (1998) Characteristics of object oriented modeling methods. In: EMISA Forum 8 (8), pp. 10–39

Nietzsche F. (1887) Zur Genealogie der Moral. Vgl. http://gutenberg.spiegel.de/buch/zur-genealogie-der-moral-3249/5

Pfänder A. (1921) Logik. Verlag von Max Niemeyer, Halle a. d. Saale

Sander H. (2011) Was genau ist eigentlich ein Modell? http://www.hsander.net/wordpress/2011/01/11/was-genau-ist-eigentlich-ein-modell/ Last Access: 31/01/2017

Schulz von Thun F. (2017) Miteinander reden: 1. Rowohlt Taschenbuch Verlag, Reinbek bei Hamburg, 54. ed.

Sloman A. (2016) What's information, for an organism or intelligent machine? How can a machine or organism mean? http://www.cs.bham.ac.uk/research/projects/cogaff/09.html$%5C#$905 Last Access: 31/01/2017

Stachowiak H. (1973) Allgemeine Modelltheorie. Springer-Verlag, Wien, New York

Stachowiak H. (1983) Erkenntnisstufen zum systematischen Neopragmatismus und zur Allgemeinen Modelltheorie. In: Modelle - Konstruktionen der Wirklichkeit. H. S. (ed.) Wilhelm Fink Verlag, p. 87

Stachowiak H. (1992) Modell. In: Handlexikon zur Wissenschaftstheorie. Seiffert H., Radnitzky G. (eds.) Deutscher Taschenbuch Verlag GmbH & Co. KG, München

Ungermann M. (2017) Was ist eigentlich ein Modell? https://arctrain.de/de/what-is-a-model/ Last Access: 31/01/2017

Yudkin J. (2012) Pure white, and deadly. Penguin Books

# Specialisation and Generalisation of Processes

Christine Choppy[a], Jörg Desel[*,b], Laure Petrucci[a]

[a] LIPN, CNRS UMR 7030, Université Paris 13, Université Sorbonne Paris Cité, France
[b] FernUniversität in Hagen, Germany

Abstract. *In conceptual data modelling, one of the most important abstraction concepts is specialisation, with generalisation being the converse. Although there are already some approaches to define generalisation for behavioural modelling as well, there is no generally accepted notion of process generalisation. In this paper, we introduce a general definition of process specialisation and generalisation. Instead of concentrating on a specific process description language, we refer to labelled partial orders. For most process description languages, behaviour (if defined at all) can be expressed by means of this formalism. We distinguish generalisation from aggregation, and specialisation from instantiation. For Petri nets, we provide examples and suggest associated notations. Our generalisation notion captures various previous approaches to generalisation, for example ignoring tasks, allowing alternative tasks and deferring choices between alternative tasks. A general guideline is that a more general process contains less features and/or less information than a more specific one.*

Keywords. Process Modelling • Process Generalisation • Process Specialisation

## 1 Introduction

Specialisation, and its counterpart generalisation, is an important concept of conceptual data modelling which has been known for many years in database research (Smith and Smith 1977). The core idea of generalisation is to combine object types, which share common attributes, to a more general supertype, which only has the common attributes, whereas each more specific type inherits these attributes from the supertype and has its own, private attributes. We can also adopt a top-down view instead of a bottom-up one: starting with an object type, we might identify subtypes such that the objects in each of the subtypes share common additional attributes, which might be meaningless for other objects. The initial type can be specialised to these subtypes, and then common attributes and additional attributes of the subtypes can be distinguished. Such a specialisation can cover the supertype (every object belongs to at least one subtype) or not, and it can divide the supertype into disjoint subtypes (no object belongs to more than one subtype) or not.

Instead of attributes of objects, generalisation and specialisation also apply to classes and their methods in object-oriented modelling, and in an even broader scope, to arbitrary features that are inherited from the more general to the more specific component. Generalisation and specialisation are very important abstraction concepts in models that clarify mutual dependencies. They are the prerequisite for reuse of system components, and they allow to avoid redundancy. For the maintenance of systems and of models, only these concepts support that common features of different system or model components can be handled at a single place, instead of considering various copies.

Whereas generalisation and specialisation are abstraction techniques that have their own graphical representation in conceptual data modelling,

* Corresponding author.
E-mail. joerg.desel@fernuni-hagen.de
This work was achieved while the second author was visiting professor at University Paris 13.

there are only few suggestions how to apply comparable concepts to process models. On the other hand, the same arguments as above would apply to a generalisation and specialisation concept in behavioural modelling as well. At several places, this demand was explicitly expressed, see e.g. (Frank and Laak 2002).

Petri nets are often proposed for conceptual modelling of system behaviour, e.g. by (Mayr et al. 2007). In more recent work (Mayr and Michael 2012), a Petri net-like language is presented for conceptual models of human behaviour. Actually, this language has primitives for operations and their pre- and post-conditions. Therefore, a natural semantics for this language is based on causal relations between operation occurrences, as occurrence nets are a well established semantics for Petri nets. Basically, each run of a Petri net model can thus be represented by a partially ordered set of occurrences of operations or transitions.

There are already some papers dealing with specialisation for particular process models, such as Petri nets. In (Wyner and Lee 2005) a particular extension of Petri nets is suggested, based on (Lee and Wyner 2003). Unfortunately, it remains unclear how this approach can be transferred to other modelling languages. Another relevant approach is given in (Aalst and Basten 2002); however, this paper also restricts to a particular language. Moreover, it emphasises inheritance and change instead of specialisation and generalisation. In particular, the inheritance is based on *blocking* and *hiding* of transitions whereas in our notion blocking a transition will turn out to be a specialisation and hiding a transition will turn out to be a generalisation.

Another important difference between our approach and previous work is that we consider partial order behaviour of process models instead of strings and sequential automata. As mentioned above, this choice is justified by the observation that many process languages emphasise causality and concurrency between activities, which is most appropriately represented by means of partial orders.

Our approach should be applicable to as many process modelling languages as possible. Therefore, we do not start with any particular syntactical description but rather concentrate on the behaviour of models. As mentioned before, the behavioural notion used in this paper is given by partially ordered sets of activities representing runs, labelled by respective tasks that are expected to appear in a process model. Although this formalism is quite easy to understand, it needs some involved technical notations that will be carefully introduced in Section 2. In our examples, we use Petri nets as a modelling language, but this does not restrict our approach to this particular language. We use only elementary Petri nets and expect the reader to understand them without any formal definition.

Our core criteria for a specialisation definition are that specialisation means to add something (features, tasks, information) to a process model and that everything valid for a more general process model should also hold for its specialisation. For example, adding a task to a process model results in the addition of respective activities in the runs, where the remaining structure of the runs is not changed. If, according to a process model, two tasks can be executed independently (in any order or concurrently), then in a specialisation an order between these tasks can be specified. This results in runs that are also runs of the more general system. If two tasks can be executed alternatively, then a specialisation might add the information to decide which of the tasks occurs; in this case, some of the runs of the more general model are ruled out in the specialisation. All these relations can be formulated by means of the respective sets of runs, and will be the subject of Section 3.

We also identify more subtle specialisation relations that refer to the branching behaviour of process models; a more special model could have "earlier" information about the decision of a choice than a more general model, although their respective sets of runs are identical. For a motivating example and our solution to this phenomenon, see Section 4.

Section 5 comes back to the previous work on specialisation of Petri nets mentioned above. We show that these concepts can be viewed as a special case of our approach.

Finally, Section 6 provides a summary and relates our notion of specialisation to other abstraction concepts such as refinement and instantiation.

## 2 Basic Setting

In this paper, no formal definition of a process model is given, since we do not stick to any particular process modelling language. Instead, some characteristics of a process model are expected to be defined: its set of tasks; its runs containing task executions; and a precedence relation between task executions in runs. This precedence relation is formally given as a partial order, i. e. a transitive and irreflexive relation (in accordance with Workflow Management Coalition n.d., we use the term activity for a single execution of a task in a process run). Our definition resembles the definition of Partial Languages as defined by (Grabowski 1981) and Pomsets as defined by (Pratt 1986).

Given a process model $P$ with a set of tasks $T$, its behaviour is defined by its set of possible runs.

**Definition 1 (Process run)** A *process run* (or just *run*) $\pi$ of a process model $P$ with set of tasks $T$ is given by

- a finite set of *activities* $A_\pi$,
- partially ordered by a *precedence relation* $\rightsquigarrow_\pi$, and
- a *mapping* $\lambda_\pi : A \to T$ mapping each activity *to a task*.

**Remark 1** We have decided to consider finite runs only because infinite runs of process models are only of theoretical interest. However, each infinite run can be approximated by an infinite sequence of finite runs, where each run is a proper prefix (defined below) of its successor.

If an activity $x$ is mapped to a task $\lambda(x)$ then it represents an occurrence of $\lambda(x)$. Activities have to be distinguished from tasks since there can be more than one occurrence of a task in one run, with different precedence relations to other activities.

**Definition 2 (Immediate precedence)**
Given a process run $\pi$, the immediate precedence relation is denoted by $\rightarrow_\pi := \rightsquigarrow_\pi \setminus (\rightsquigarrow_\pi \circ \rightsquigarrow_\pi)$.

**Remark 2** Since the set of activities $A_\pi$ is finite, $\rightsquigarrow_\pi$ is the transitive closure of $\rightarrow_\pi$.

In graphical representations, we depict the relation $\rightarrow_\pi$ by means of directed arcs, i.e., we provide the Hasse-diagram of partial orders. Two activities $x$ and $y$ satisfy $x \rightsquigarrow_\pi y$ if and only if there is a nonempty path leading from $x$ to $y$.

**Example 1**
Figure 1 depicts a process run $\pi$ such that : $A_\pi = \{1, 2, 3, 4, 5\}$, $\rightarrow_\pi = \{(1, 2), (2, 3), (3, 4), (1, 5)\}$, and $\lambda_\pi(1) = a$, $\lambda_\pi(2) = b$, $\lambda_\pi(3) = c$, $\lambda_\pi(4) = b$, $\lambda_\pi(5) = b$. Thus, activities are denoted by numbers and tasks by lowercase letters.

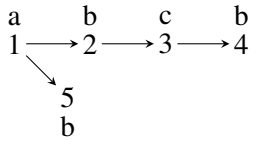



*Figure 1: A process run π*

Note that a process run can feature several branches and that a given task can occur at different places in the process run. In the above example, task b can occur after another occurrence of task b, and both occur concurrently to a third one. However, they are associated with different activities, respectively 4 and 2, which are ordered by the precedence relation, and 5.

In case of sequential semantics of processes, where each run is represented by a sequence of occurring tasks, we can easily distinguish the first, the second, etc. occurrence of a single task. Each sequence can be viewed as a mapping from the set $\{1, 2, 3, \ldots, length(run)\}$ to the set of tasks. For partial-order semantics, there is no such

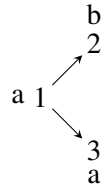
unique representation of a run. Hence the "same behaviour" can be represented by runs which only differ w.r.t. the activities. Such runs are said to be isomorphic.

**Definition 3 (Isomorphic runs)** Let $\pi$ and $\pi'$ be two process runs of the same process model $P$ with set of tasks $T$. Then, $\pi'$ is *isomorphic* to $\pi$ if there is a bijection $\beta\colon A_\pi \to A_{\pi'}$ such that

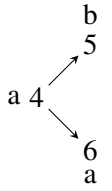
1. $\forall x, y \in A_\pi\colon\ x \rightarrow_\pi y \iff \beta(x) \rightarrow_{\pi'} \beta(y)$ and
2. $\forall x \in A_\pi\colon\ \lambda_\pi(x) = \lambda_{\pi'}(\beta(x))$.

Clearly, process run isomorphism is an *equivalence relation*. If isomorphic process runs are not distinguished, any representative of the equivalence class can be used.

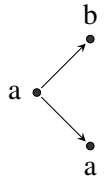
**Example 2** Figure 2 schematises the isomorphism between two process runs. Process run $\pi$ (Figure 2(a)) and process run $\pi'$ (Figure 2(b)) yield the same graph of tasks when abstracting away the activities.




(a) Process run $\pi$




(b) Process run $\pi'$




(c) Task occurrences

*Figure 2: A process run $\pi'$ isomorphic to a process run $\pi$*

In general, activities are formalised differently even though they have the same meaning in terms of tasks. Sometimes it is still necessary to distinguish activities within a process run in order to be able to refer to precise occurrences of a task. When this distinction is irrelevant in a figure, a $\bullet$ will be used instead of the actual activity (see e.g. Figure 2(c)).

## 3 Linear Time Specialisation

The meaning of $a \rightsquigarrow_\pi b$ is that $\lambda(a)$ is executed before $\lambda(b)$ in run $\pi$. If activities $a$ and $b$ are not ordered by means of $\rightsquigarrow_\pi$ then they occur in any order (this order is not captured by our notion of run) or concurrently. Each *linearisation* of a run $\pi$, obtained by adding elements to the order relation $\rightsquigarrow_\pi$, yields another run, which we consider more *specific* than $\pi$. In turn, each run *generalises* its set of linearisations.

The following definition not only compares the precedence relations between activities but also the respective sets of activities of two runs. It formalises the observation that a run of a more specific process definition might contain more details than a run of a less specific process definition. Hence the runs in the following definition can belong to distinct processes.

**Definition 4 (Process run specialisation)** Let $P$ and $P'$ be two process models with sets of tasks $T$ and $T'$, respectively. A process run $\pi'$ of $P'$ *specialises* a process run $\pi$ of $P$ (denoted by $\pi' \geq \pi$) if there is an injective mapping $\mu\colon A_\pi \to A_{\pi'}$ such that

1. $\forall x, y \in A_\pi\colon\ x \rightarrow_\pi y \Rightarrow \mu(x) \rightsquigarrow_{\pi'} \mu(y)$ and
2. $\forall x \in A_\pi\colon \lambda_\pi(x) = \lambda_{\pi'}(\mu(x))$.

**Remark 3** As a consequence of Definition 4, if $x \rightsquigarrow_\pi y$ then $\mu(x) \rightsquigarrow_{\pi'} \mu(y)$.

Each activity in the process run $\pi$ has a corresponding activity in the specialised process (by the

mapping $\mu$), mapped to the corresponding task (2). Moreover, for each precedence relation in $\pi$, there is a corresponding sequence of precedences in $\pi'$ (1).

**Example 3** Figure 3 shows a process run $\pi'$ specialising a process run $\pi$ together with the mapping $\mu$.

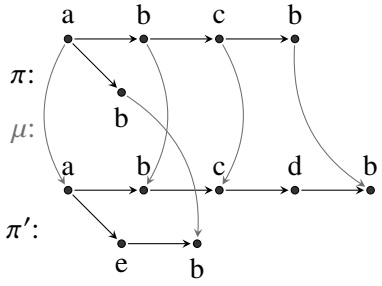



*Figure 3: Process run $\pi'$ specialises process run $\pi$*

Up to an isomorphism, a process run $\pi'$ of $P'$ specialises a process run $\pi$ of $P$ if and only if $A_\pi \subseteq A_{\pi'}$ and $\rightsquigarrow_\pi \subseteq \rightsquigarrow_{\pi'}$. This justifies the "$\geq$"-notion for specialisations. It is formalised by the following lemma.

**Lemma 1** *Let $P$ and $P'$ be two process models with sets of tasks $T$ and $T'$, and runs $\pi$ and $\pi'$, respectively. We have $\pi' \geq \pi$ if and only if there exists a process run $\overline{\pi}$ such that*

1. *$\overline{\pi}$ is isomorphic to $\pi$,*
2. *$A_{\overline{\pi}} \subseteq A_{\pi'}$,*
3. *$\rightarrow_{\overline{\pi}} \subseteq \rightsquigarrow_{\pi'}$, and*
4. *$\forall x \in A_{\overline{\pi}} : \lambda_{\overline{\pi}}(x) = \lambda_{\pi'}(x)$.*

Proof ($\Rightarrow$) Assume that $\pi'$ specialises $\pi$ by means of mapping $\mu$. Process run $\overline{\pi}$ is constructed as follows:

$A_{\overline{\pi}} = \{x' \in A_{\pi'} \mid \exists x \in A_\pi : x' = \mu(x)\}$;

$\rightsquigarrow_{\overline{\pi}} = \{(x', y') \in A^2_{\overline{\pi}} \mid \exists (x, y) \in \rightsquigarrow_\pi : \mu(x) = x' \wedge \mu(y) = y'\}$;

$\forall x \in A_{\overline{\pi}} : \lambda_{\overline{\pi}}(x) = \lambda_{\pi'}(x)$.

($\Leftarrow$) Assume that $\overline{\mu} : A_{\overline{\pi}} \rightarrow A_{\pi'}$ is an isomorphism. Mapping $\mu : A_\pi \rightarrow A_{\pi'}$ is constructed by: $\forall x \in A_\pi, \mu(x) = \overline{\mu}(x)$.

A process run $\pi'$ specialises $\pi$ if $\pi$ boils down to the same tasks graph as process $\overline{\pi}$ (4), $\overline{\pi}$ is isomorphic to $\pi$ (1), but with activities in $\pi'$ (2). The precedence relation in $\overline{\pi}$ respects the order relation in $\pi'$ (3).

**Definition 5 (Linear time specialisation)** A process model $P'$ is a *linear time specialisation* of a process model $P$ if for each run $\pi'$ of $P'$, there exists a run $\pi$ of $P$ such that $\pi' \geq \pi$.

The process model $P$ is then said to be a *a linear time generalisation* of $P'$.

As mentioned in the introduction, every valid statement about the more general process should hold for the specialised process as well. In a more formal setting, which is beyond the scope of this paper, any formula of an appropriate logic that evaluates to *true* for the general process, should be evaluated to *true* for the specialised process as well. Since, in this section, we concentrate on a relation based on the runs only, this logic would be Linear Time and based on partial orders, such as the one of (Bhat and Peled 1998). The idea is that a less general process contains more features/information than a more general one but inherits all properties from the more general one.

## Representative examples of specialisation/generalisation:

Table 1 depicts some representative examples of generalisation and specialisation. The process models are given as Petri nets, and their runs are pictured as well. For each example, both a specialised and a general version are given. The example is named after the specialisation characteristic. Hence, the first one *forces an activity* since it prevents an activity associated with task b from occurring. The second example *adds an activity* associated with task b. Finally, the last example imposes a sequential *ordering between activities* that are concurrent in the general model, namely those associated with tasks b and c.

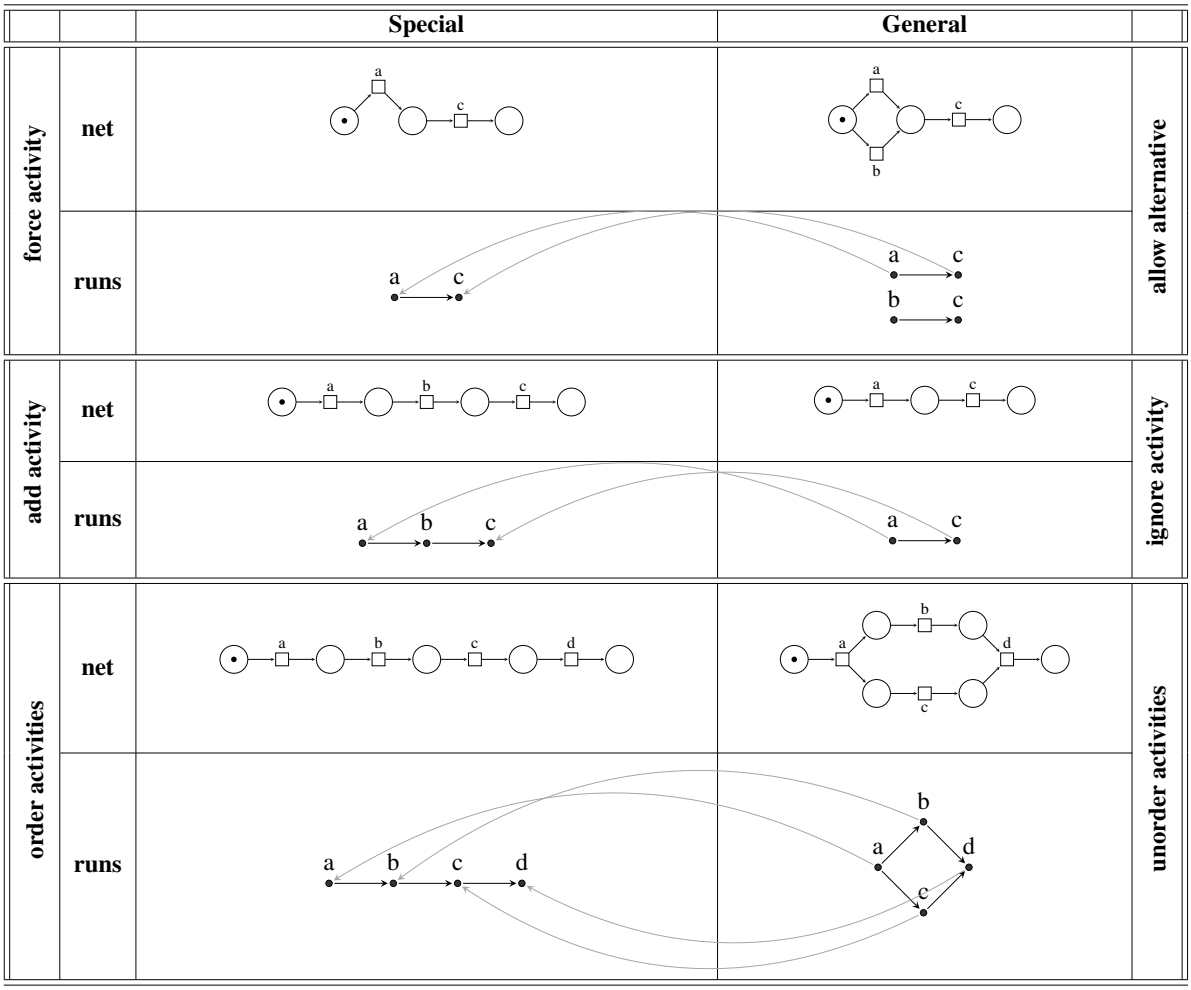



*Table 1: Representative examples of specialisation/generalisation*

## 4 Branching Time Specialisation

We begin this section with an example motivating further enhancement of the specialisation notion.

**Example 4** Consider two processes $P$ and $P'$ modelled by the Petri nets in Table 2. These nets are actually labelled Petri nets. The two transitions of the net on the left-hand side labelled by $a$ represent the same task $a$. As shown in the pictures, they have identical sets of runs. Both processes start with an occurrence of $a$ and then either continue with an occurrence of $b$ or with an occurrence of $c$. Hence they cannot be distinguished by just inspecting their runs. However, in process $P$, after the occurrence of $a$, there is a choice between continuing with $b$ or with $c$, whereas in process $P'$ we have to choose immediately between the run containing occurrences of $a$ and $b$, and the one containing occurrences of $a$ and $c$. In this case we might consider process $P'$ as a specialisation of $P$ since additional (more precisely, earlier) information about the choice between $b$ and $c$ is necessary.

We cannot distinguish processes $P$ and $P'$ by means of their runs as in the previous section. Therefore, it is necessary to carefully examine all possible behaviours. To do so, we first define the prefix of a run.

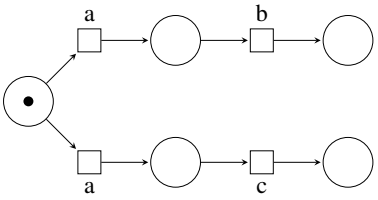
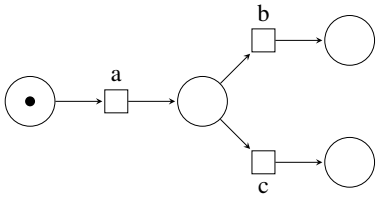
| | P' | P |
|---|---|---|
| **net** |   |   |
| **runs** |   |   |

*Table 2: Runs do not always distinguish processes*

**Definition 6 (Prefix of a run)** A run $\pi$ is a *prefix of a run* $\pi'$ if there is an injective mapping $\mu\colon A_\pi \to A_{\pi'}$ such that

1. $\forall x, y \in A_\pi, x \to_\pi y \iff \mu(x) \to_{\pi'} \mu(y)$,
2. $\forall x \in A_\pi, \lambda_\pi(x) = \lambda_{\pi'}(\mu(x))$, and
3. $\forall x', y' \in A_{\pi'} : x' \to_{\pi'} y' \Rightarrow [(\exists y \in A_\pi, y' = \mu(y)) \Rightarrow (\exists x \in A_\pi, x' = \mu(x))]$.

**Explanation:**
Each precedence relation in the prefix has a correspondence in the complete run $\pi'$, (1) and corresponding activities are mapped to the same task (2). Moreover, each precedence relation of the complete run $\pi'$ leading to an activity that corresponds to one of the prefix also has its starting point corresponding to an activity of the prefix (3).

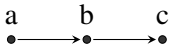
**Example 5** The process run of Figure 4(a) is a prefix of the process runs of Figures 4(b) and 4(c).

Roughly speaking, a prefix of a run is constituted by a set of activities together with all their predecessors, up to an isomophism. This observation is formalised in the following lemma:

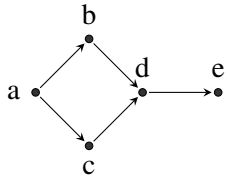
**Lemma 2** *A run $\pi$ is a prefix of a run $\pi'$ if and only if there exists a process run $\overline{\pi}$ such that*

*1. $\overline{\pi}$ is isomorphic to $\pi$,*

(a) Process run $\pi$



(b) Continuation $\pi'$



(c) Continuation $\pi''$

*Figure 4: A process run $\pi$ with different extended runs $\pi'$ and $\pi''$*

*2. $A_{\overline{\pi}} \subseteq A_{\pi'}$,*
*3. $\to_{\overline{\pi}} = \to_{\pi'} \cap A_{\overline{\pi}} \times A_{\overline{\pi}}$,*
*4. $\forall x \in A_{\overline{\pi}}, \lambda_{\overline{\pi}}(x) = \lambda_{\pi'}(x)$,*
*5. $\forall y \in A_{\overline{\pi}}, x \to_{\pi'} y \Rightarrow x \in A_{\overline{\pi}}$.*

Proof ($\Rightarrow$) Let $\mu$ be the mapping mentioned in the definition of a prefix. Set $A_{\overline{\pi}} = \{x' \in A_{\pi'} \mid \exists x \in A_\pi : x' = \mu(x)\}$. The other defining components of $\overline{\pi}$ are items 3 and 4 of the lemma. ($\Leftarrow$) Choose $\mu(x) = \mu_i(x)$ for every $x \in A_\mu$ where $\mu_i$ is the isomorphism from $\pi$ to $\overline{\pi}$.

**Explanation:**
This lemma is very similar to Lemma 1. Run $\overline{\pi}$ is exactly the restriction of run $\pi'$ to the activities in $\overline{\pi}$.

Now, for a run that is a prefix of another run, we define the set of its continuations.

**Definition 7 (Extended run, continuations)**
An *extended run* is a run $\pi$ together with a set of runs $C(P)$, called its *continuations*, such that $\pi$ is a prefix of each run in $C(P)$.

Note that in general $C(P)$ does not contain all runs with prefix $\pi$. Two extended runs are different if their continuations are different, even if their respective runs are isomorphic.

**Example 6** Figure 4 pictures a process run $\pi$ (Figure 4(a)), and two different continuations: $\pi'$ in Figure 4(b) and $\pi''$ in Figure 4(c). Then $(\pi, \{\pi'\})$ and $(\pi, \{\pi''\})$ are two different extended runs of $\pi$.

Isomorphic runs can lead to different states. Definition 7 avoids considering states of processes. For applying our concept to concrete process definition languages, we might consider the states reached by runs instead, determining the possible continuations.

**Remark 4** For unlabelled Petri nets the distinction does not occur because isomorphic runs lead to identical markings. For labelled Petri nets the problem can occur.

**Definition 8 (Branching Time Specialisation)**
A process model $P'$ is a *branching time specialisation* of a process model $P$ if, for each extended run $\pi'$ of $P'$ with continuation $C(P')$, there exists an extended run $\pi$ of $P$ with continuation $C(P)$ such that

- $\pi' \geq \pi$, and
- for each run $\tilde{\pi}' \in C(P')$ there exists a run $\tilde{\pi} \in C(P)$ such that $\tilde{\pi}' \geq \tilde{\pi}$.

**Example 7** The nets in Table 2 can be distinguished using the specialisation of Definition 8, while they could not be with the linear time specialisation of Definition 5.

In Table 3, the activities are numbered after transitions names. The process model $P'$ has a run $\pi'$, consisting only of activity 1 labelled by $a$. Its set of continuations $C(P')$ contains this run itself and also the run $1 \rightarrow 2$, as given in the figure. For $P$, we also have the run $\pi = 1$, labelled by $a$. Its set of continuations $C(P)$ contains also the run $1 \rightarrow 4$, as given in the figure. So, in this example, for every continuation in $C(P')$ we find an isomorphic continuation in $C(P)$. Conversely, consider the extended run $\pi$ consisting only of activity 1 labelled by $a$ in process model $P$, which is a prefix of both depicted runs. Its continuation is the set of both runs depicted in the table. So it is possible to continue with a $b$-activity and it is also possible to continue with a $c$-activity. Now activity 1 (i.e., the run consisting only of this activity) can not be an associated run of $P'$ because it misses a continuation with an activity labelled with $c$. Similarly, activity 3 can not be an associated run of $P'$ because this one misses a continuation with an activity labelled with $b$. Arguing about all possible runs and continuations of $P'$ as above shows that $P'$ is a specialisation of $P$.

We call this concept of specialisation Branching Time Specialisation because, again, we have that the valid statements (now expressable in Branching Time Temporal Logic) of the more general process should also hold for the specialised one.

## 5 Related Works

Abstraction and refinement have been the subject of numerous researches, and we just mention some of them here. The goals include to keep the modelling and reasoning as close as possible to the essence of the system to be developed, while still taking into account that a concrete representation (also called concrete implementation) of data should be provided at some later point together with the way it is related with the abstract data.

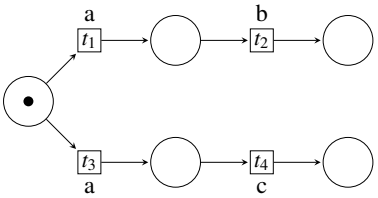
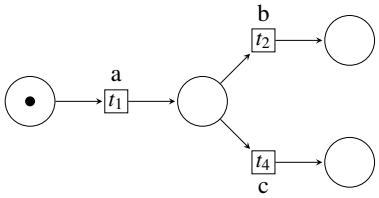
| | P' | P |
|---|---|---|
| **net** |  |  |
| **runs** | a b<br>1 → 2<br>a c<br>3 → 4 | a b<br>1 → 2<br>a c<br>1 → 4 |

*Table 3: Runs display different activities*

(Hoare 1972) proposed a method to prove program correctness in the context of stepwise refinement where a representation of abstract data is chosen. Later, (Guttag et al. 1978) showed how the use of algebraic axiomatizations can simplify the process of proving the correctness of an implementation of an abstract data type. A number of works followed in the algebraic specification field. Now, an explicit data type refinement construct is introduced in programming languages like Scala (The Scala Programming Language n.d.). Refinement can also be used for program derivation as in (Diallo et al. 2015), who define a correctness based concept of refinement.

As mentioned in the introduction, abstraction was studied for database modelling in (Smith and Smith 1977), with the concepts of generalisation and aggregation. These concepts are also part of the object-oriented approaches (e.g. UML based approaches), or languages.

The same concepts should also be adapted to the modelling of the behaviour of a system, described by a process, a state-based diagram, etc.

(Lee and Wyner 2003) define a specialisation concept for dataflow diagrams. They distinguish minimal execution set semantics, in which adding an activity is a specialisation (as in Table 1), and the maximal execution set semantics, which is the option they choose. So obviously, their definition of specialisation differs from ours.

In (Wyner and Lee 2005), they work on a specialisation for a variant of Petri net (Workflow Process Definition). They analyse the approach of (Aalst and Basten 2002) who identify four types of inheritance of workflows, and propose an extension.

Several Petri nets refinements (node, type, subnet) were defined by (Lakos 2000), and later extended in (Choppy et al. 2013) to propose type refinement that complies with the subtype relation defined by (Liskov and Wing 1994). These refinements are useful both, for an incremental development of the specification, and for substantial gain in model checking properties.

(Wang et al. 2010) stress that design issues are important, and, in the context of component-based model-driven development, they present two refinement relations, a trace-based refinement and a state-based (data) refinement, that provide different granularity of abstractions. So they combine the data refinement and the behaviour refinement.

Along similar lines (Fayolle et al. 2015) propose to couple a dynamic behaviour specification expressed with ASTD (Algebraic State Transition Diagrams) with a data model described by means of an Event-B specification. The complementarity and consistency of refinements for both parts

are explored, and the approach is illustrated on a CBTC-like train controller.

## 6 Conclusions

### Summary

We have presented a very general notion of specialisation and generalisation of processes which does not stick to a specific process modelling language but can be applied to all process languages where behaviour can be expressed in terms of partially ordered activities. Processes with sequential runs are a special case where all activities in runs are mutually ordered. Specialisation is interpreted as addition of features, where features can be additional tasks, additional ordering information, additional information on choices and earlier information on choices. All these features except the last one can be expressed by a specialisation relation on runs whereas the last one concerns branching points of process models that do not appear in runs and hence require a more involved definition based on runs and their possible continuations.

### Specialisation versus Refinement/Generalisation versus Aggregation

One could argue that refinement of a process element is a form of specialisation because the more detailed view adds information. Conversely, what distinguishes generalisation and aggregation? According to our definition, specialisation *adds* something to a process, whereas refinement *replaces* something by something else, which should be more detailed. For example, if a task is added to the process, then the process is more special and associated activities show up in its runs. If a task is refined to two subsequent tasks, then the process is more detailed and the respective activities are refined accordingly in its runs.

### Specialisation versus Instantiation

In one of our previous examples we have shown that information about the decision of choices can be viewed as a particular specialisation. The specialised process contains less alternatives. If all choices are decided, then the process is deterministic and contains no alternative at all. In other words, it only has a single run (remember that our notion of a run captures possible concurrency so that no additional runs caused by interleaving appear). Depending of the representation of the process and of its single run, both might look very similar. However, the run represents an instance of the process (and was an instance of the original process, too) whereas the process is on a lower meta-level. Whereas repeated specialisation of processes is possible and always yields new processes, instantiation of processes decreases the meta-level by one and can only occur once.

### Extensions

It is obvious that there are features, that can be added, which cannot be handled by our concept so far. One example concerns concurrency. Our notion of partially ordered runs is interpreted in such a way, that activities that are not ordered can occur concurrently or in any order. However, it could be possible to specialise such a specification by demanding that activities have to occur concurrently whereas any order would be illegal. This cannot be expressed by our notion unless we add an additional "concurrent" relation. Other, similar relations are "not later than" or "at most two of three activities occur concurrently".

Another extension refers to data. Usually tasks are performed for some or several data objects. This data does not appear in runs of our processes. For a combined view of processes and data, and for a combined notion of specialisation, the process view and the data view have to be integrated. There is an obvious way to do this integration by carrying over the data attributes from tasks to activities. Then the mapping between activities constituting our specialisation relation has to be extended to these data objects; the more general process handles the more general data.

### Representation

In data modelling, generalisation and other abstraction techniques such as aggregation are represented graphically in the model. While this is

nicely possible for aggregation in process models (see, for example, Desel and Merceron 2010), we do not see an elegant solution for generalisation in process models. One obvious approach is to depict additional tasks and additional relations between tasks by means of different symbols (colours or lines, respectively). However, if a single specialisation considers changes in different parts of a process then it is not obvious to express that these changes can only occur together.

## The Partial Order of Specialisation

Our notion of generalisation and specialisation is not primarily given as a problem (is $x$ a specialisation of $y$ ?) but as a specification means. However, it is an interesting question whether the above question is decidable and, in the positive case, what is the complexity of a decision algorithm or of the problem in general (i. e., the complexity of the most efficient algorithm).

It is not difficult to see that the specialisation relation between single process runs is decidable, although any algorithm heavily depends on the data structure used to represent the respective process runs. Notice that by definition process runs are finite, i. e., have a finite set of activities. We cannot deal with isomorphism classes of process runs in algorithms but instead consider arbitrary representative process runs.

**Proposition 1** *Given two process runs $\pi$ and $\pi'$ of two process models $P$ and $P'$, it is decidable whether $\pi'$ specialises $\pi$.*

Proof $\pi'$ can only be a specialisation of $\pi$ if, for each task $t$, $\pi'$ has at least as many activities labelled by $t$ as $\pi$ has. This condition is easy to check. Assume that it holds true.

There are finitely many injective label-preserving mappings from the activities of $\pi$ to the activities of $\pi'$, and it is not difficult to construct them in a systematic way. For each pair of activities of $\pi$, which are in the immediate precedence relation, we check whether the respective target activities are ordered in $\pi'$ (they do not necessarily have to be immediate successors!). If this is true for at least one mapping, then $\pi'$ is a specialisation of $\pi$.

Things become more difficult when process models are compared because each process model can have infinitely many runs. Since we do not consider any particular process model, nothing can be said about decidability in general. Clearly, there is only a chance for decidability of specialisation if a process model cannot generate infinitely many arbitrary runs.

It is not difficult to prove that "being a specialisation" is a partial order on process models. Its minimal element is the empty process model which is a (useless, though) generalisation of all process models. Using this order one might ask, for a given set of process models, whether there is a "largest" generalisation, whether this is unique, and whether it can be constructed algorithmically. Similarly, it would be interesting to find the "smallest" specialisation of a set of process models. There are no obvious answers to these questions, and hence the study of the partial order of specialisation is subject to future work.

## References

van der Aalst W. M. P., Basten T. (2002) Inheritance of workflows: an approach to tackling problems related to change. In: Theoretical Computer Science 270(1-2), pp. 125–203

Bhat G., Peled D. (1998) Adding Partial Orders to Linear Temporal Logic. In: Fundamenta Informaticae 36(1), pp. 1–21

Choppy C., Petrucci L., Sanogo A. (2013) Coloured Petri Nets Refinements. In: Proceedings of the workshop on Petri Nets and Software Engineering (PNSE'13), Workshop Proceedings 989, CEUR, pp. 187–201

Derrick J., Boiten E. A., Reeves S. (eds.) Proceedings 17th International Workshop on Refinement, RefineFM 2015. EPTCS Vol. 209 https://doi.org/10.4204/EPTCS.209

Desel J., Merceron A. (2010) Vicinity Respecting Homomorphisms for Abstracting System Requirements. In: Transactions on Petri Nets and Other Models of Concurrency Lecture Notes in Computer Science 4 Jensen K., Donatelli S., Koutny M. (eds.), pp. 1–20

Diallo N., Ghardallou W., Desharnais J., Mili A. (2015) Program Derivation by Correctness Enhacements. In: Derrick J., Boiten E. A., Reeves S. (eds.) Proceedings 17th International Workshop on Refinement, RefineFM 2015, Oslo, Norway, 22nd June 2015.. EPTCS Vol. 209, pp. 57–70 https://doi.org/10.4204/EPTCS.209.5

Fayolle T., Frappier M., Laleau R., Gervais F. (2015) Formal refinement of extended state machines. In: Derrick J., Boiten E. A., Reeves S. (eds.) Proceedings 17th International Workshop on Refinement, RefineFM 2015, Oslo, Norway, 22nd June 2015.. EPTCS Vol. 209, pp. 1–16 https://doi.org/10.4204/EPTCS.209.1

Frank U., Laak B. V. (2002) A Method for the Multi-Perspective Design of Versatile E-Business Systems. In: Proceedings of the 8th Americas Conference on Information Systems, pp. 621–627

Grabowski J. (1981) On partial languages. In: Fundamenta Informaticae 4(2), pp. 428–498

Guttag J. V., Horowitz E., Musser D. R. (1978) Abstract data types and software validation. In: Commun. ACM 21(12), pp. 1048–1064

Hoare C. A. R. (1972) Proof of Correctness of Data Representations. In: Acta Informatica 1, pp. 271–281

Lakos C. (2000) Composing Abstractions of Coloured Petri Nets.. In: Nielsen, M., Simpson, D. (eds.) 21st Int. Conf. on Application and Theory of Petri Nets (ICATPN). Lecture Notes in Computer Science Vol. 1825. Springer-Verlag, pp. 323–345

Lee J., Wyner G. M. (2003) Defining specialization for dataflow diagrams. In: Information Systems 28(6), pp. 651–671

Liskov B. H., Wing J. M. (1994) A Behavioral Notion of Subtyping. In: ACM Trans. on Programming Languages and Systems 16(6), pp. 1811–1841

Mayr H. C., Kop C., Esberger D. (2007) Business Process Modeling and Requirements Modeling. In: First International Conference on the Digital Society (ICDS'07), pp. 8–8

Mayr H. C., Michael J. (2012) Control pattern based analysis of HCM-L, a language for cognitive modeling. In: International Conference on Advances in ICT for Emerging Regions (ICTer2012). IEEE, pp. 169–175

Pratt V. (1986) Modelling Concurrency with Partial Orders. In: Int. Journal of Parallel Programming 15, pp. 33–71

Smith J. M., Smith D. C. P. (1977) Database abstractions: aggregation and generalization. In: ACM Trans. Database Systems 2(2), pp. 105–133

The Scala Programming Language (n.d.) http://www.scala-lang.org/. Last Access: Accessed on: 29.1.2018

Wang Z., Wang H., Zhan N. (2010) Refinement of Models of Software Components. In: Proceedings of SAC'10, pp. 2311–2318

Workflow Management Coalition (n.d.) http://www.wfmc.org/. Last Access: Accessed on: 29.1.2018

Wyner G. M., Lee J. (2005) Applying Specialization to Petri Nets: Implications for Workflow Design. In: Bussler C., Haller A. (eds.) Business Process Management Workshops Vol. 3812, pp. 432–443

# A Petri net-based View on the Business Process Life-Cycle

Agnes Koschmider*,[a], Andreas Oberweis[a], Wolffried Stucky[a]

[a] Institute of Applied Informatics and Formal Description Methods, KIT, 76137 Karlsruhe, Germany

Abstract. *During the last 30 years Petri nets have shown a continuous popularity and durability as process modelling language. While some process modelling languages were crossed with others or even disappeared, Petri nets are continuously used as modelling language addressing various purposes in the context of business processes. In this paper, we refer to the success of Petri nets and describe business process modelling extensions as well as approaches for process modelling, simulation, execution and evaluation relying on Petri nets. The variety of Petri net-based extensions shows that Petri nets can be adapted to changing requirements for which these extensions, modifications or variants have been proposed.*

Keywords. Petri Nets • Modelling Language • Modelling Variants

## 1 Introduction

Process modelling is an important activity that supports all phases of the business process life-cycle, from the early identification of the need for a business process through subsequent phases of design, execution and monitoring. Business process models support the common understanding of the kind of business activities underlying a business process, including their mutual relationships. Plenty of modelling languages exist to design business processes with different representations (e.g., text vs. graphic) or using a different design paradigm (e.g., imperative vs. declarative). The decision in favour of a specific modelling language depends on the requirements that the chosen modelling language should meet (Oberweis 1996). Possible requirements might be the appropriateness and adequacy of modelling constructs (i. e., expressiveness) or an easy learnability of the language (i. e., simplicity). Certainly, the decision in favour of a process modelling language might also be made based upon its practical acceptance, case studies or the cognitive effectiveness of the visual notations (Figl and Recker 2016). Currently, the Business Process Model and Notation (BPMN) is mostly used for process modelling although the language has shortcomings with respect to a comprehensive support of the whole business process lifecycle (Radloff et al. 2015). The Event-Driven Process Chain (Nüttgens and Rump 2002), which was for quite long time the standard language for business process modelling, has lost popularity. Retrospectively, a continuous durability as a process modelling language can be attested only to Petri nets (Reisig 2013). This certainly can be explained due to the *"mathematical analysis techniques allowing for analytical verification of many relevant properties of systems' behaviour"* (Desel et al. 1998).

Since the doctoral thesis of Carl Adam Petri entitled "Communication with Automata" (Petri 1962) in 1962 there exist also numerous variants for Petri nets addressing various purposes in the context of business processes or system dynamics in general (e. g., different variants of high-level Petri nets). In this paper, we refer to the success of Petri nets and describe business process modelling extensions or approaches to process modelling, simulation, execution and monitoring relying on Petri nets. We will show that Petri nets are capable of expressing changing requirements for which

* Corresponding author.
E-mail. agnes.koschmider@kit.edu

these extensions or variants of Petri nets have been proposed.

The paper is structured as follows. Section 2 summarizes common languages for business process modelling and sketches their current status as modelling language. Section 3 presents exemplary extensions of Petri nets or approaches relying on Petri nets and classifies them according to the business process life-cycle. In each phase we also point to future directions of process modelling, execution and evaluation. The paper ends with a summary in Section 4.

## 2 Business Process Modelling Languages

The most common language to design business processes are the Business Process Model and Notation (BPMN), Event Driven Process Chains (EPC), Petri nets and the Unified Modelling Language (UML). Technical issues of business process models are rather tackled by orchestration and choreography languages such as XML Process Definition Language (XPDL), the Web Services Business Process Execution Language (WS-BPEL), ebXML or WS-CDL.

**BPMN** is a graphical modelling language that does not only support business process modelling but also process implementation. Accordingly, the modelling language has primarily been designed with the intention to make the notation easy to understand for all user groups (e.g., analysts creating the first drafts or the technical developers responsible for the implementation). BPMN is divided into five basic categories: Flow Objects, Data, Connecting Objects, Swimlanes and Artefacts. Figure 1 shows a BPMN diagram with three different flow objects, which are events (circles), activities (rectangles) and gateways. The standardization by the OMG in 2005 has certainly led to the high popularity of BPMN and has contributed to BPMN becoming the de-facto standard for business process modelling.

**EPC** is a graphical, semi-formal modelling language aiming to provide a simple and intuitive way to visualize business process models. There are numerous proposals in the literature formalizing

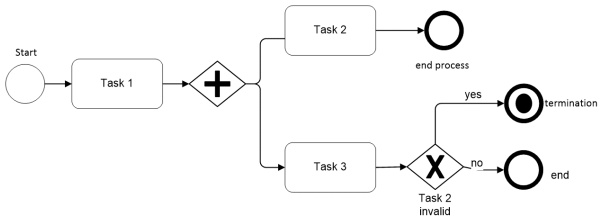



*Figure 1: Example BPMN diagram*

the EPC language (Kindler 2006; Mendling and Aalst 2006). EPC received a widespread adoption after their integration with the SAP reference model (Nüttgens and Rump 2002) and due to its part of the so-called ARIS house. The main constructs of an EPC are events, functions, and join operators (XOR, AND, OR). Numerous variants of the EPC can be found in the literature such as the extended EPC (eEPC) allowing to define organizational units and input and output parameters. Figure 2 visualizes an eEPC diagram.

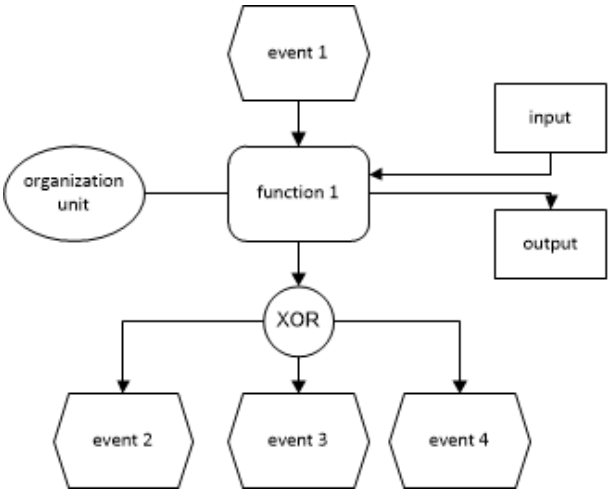



*Figure 2: Example eEPC diagram*

Due to the lack of standardization through an appropriate organization the popularity of EPCs recently significantly decreases (Karhof et al. 2016).

**Petri nets** are used in many ways today, for example to model, simulate, analyse and monitor

business processes. In doing so, sequential, mutually exclusive and concurrent activities can be modelled with three modelling elements. Petri nets consist of places, which are represented as circles, and of transitions, which are visualized as rectangles, see Figure 3. Places and transitions are linked by arcs representing relationships between both elements (Reisig 2013). Places represent static aspects, such as documents, data or resources. Transitions are used to represent dynamic aspects (i. e., state transitions). Petri nets combine the advantages of a simple graphical representation of business processes with the formal semantics of the uses notation.

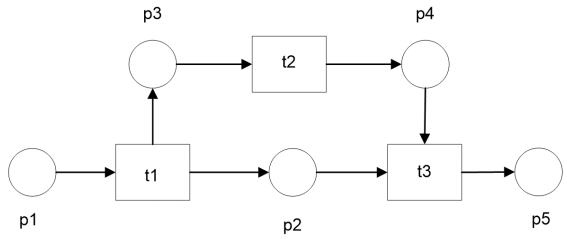



*Figure 3: Example Petri net diagram*

There have already been many discussions about the theoretical investigations of Petri nets in the academic sector and the practical needs of business process modelling languages in business applications (Desel et al. 1998). In spite of all these discussions Petri nets are the only language that "survived" the last 50 years of modelling and thus a continuous popularity can be definitely attested to Petri nets (Reisig 2013). One variant of Petri nets, which is capable to represent e-Business or web service based aspects are XML nets. XML nets are a variant of high-level Petri nets (Lenz and Oberweis 2003). They have formal semantics, graphical nature, and the strength in exchanging XML-based structured data. In XML nets markings of the places are given as XML documents and places are considered as containers for the documents. The flow of XML documents is defined through occurrences of transitions.

## 3 Extensions and Approaches relying on Petri nets

In the following we discuss extensions and approaches relying on Petri nets according to the business process lifecycle (Weske 2012) (see Figure 4). The *design & analysis* phase tackles the creation and validation of business process models. The *configuration* phase within the lifecycle is about technical realization of a process model, including setting specific parameters to tie implementation details to the higher-level model. The *enactment* phase is related to the monitoring of process models. In the business process lifecycle the *evaluation* phase addresses the evaluation of process models using business activity monitoring and process mining techniques.

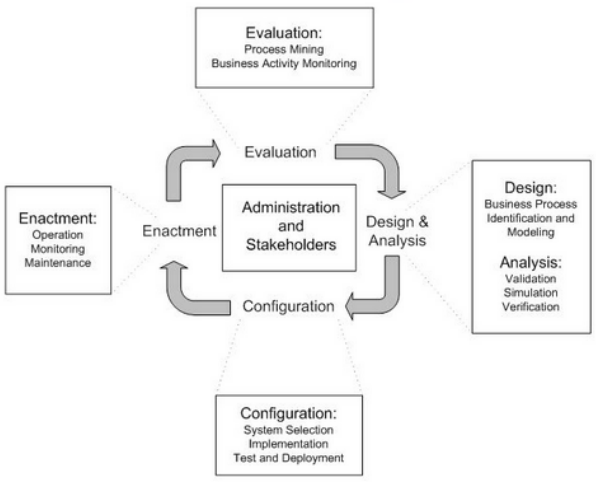



*Figure 4: Business process lifecycle*

Process mining refers to a set of techniques that analyze (mostly historical) event logs of IT systems in order to derive a model of the process (i. e., mostly a Petri net) that corresponds to these logs. For a process model to be created from such log files, the event log must (at least) store the sequence of events. Figure 5 shows a log file with two traces to which process mining algorithms were applied in order to derive a Petri net (Drescher et al. 2017). Consequently, one can observe that Petri nets also play an important role in the evolving field of process mining.

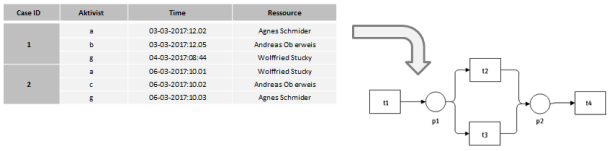



*Figure 5: Process mining applied: generating a Petri net from an event log*

The next section summarizes Petri net based approaches related to the design and analysis phase.

### 3.1 Extensions for the Design and Analysis Phase

There are many discussions regarding the modelling approach (i. e., imperative versus declarative modelling) (Fahland et al. 2009), the support of modellers and automatic creation of process models from log files (i. e., process mining). With respect to modelling methods and in particular the understandability of Petri net based models, research has been conducted on how novice modellers are constructing process models. In this area, it is well known from empirical studies that novice modellers struggle to create "good" process models since they tend to forget important model elements. One approach that supports modellers aims at reducing the entry barrier to process modelling by providing an on top layer to process models (Koschmider et al. 2015). It is a lightweight approach to modelling. Particularly, the approach presented in (Koschmider et al. 2015) discusses variables of how to best design diagrams consisting of graphical or textual elements The approach adds an abstract model level ("Layer 0") that can be represented both depictively (iconic) or descriptively (symbolic) with the possibility to seamlessly switch between them. The abstract layer should enable a quick and comprehensive view of the underlying (Petri net-based) process model and in addition should expose basic modelling capabilities. With this layer modelling should be accessible for a larger audience. From the viewer's perspective: both representations allow the process model's viewer to get a quick and comprehensive view of the underlying process model. If the viewer is further interested in the fine-grained representation of the process model he/she can navigate through the process model hierarchy. From the creator's (i. e., process modeller's) perspective: the concepts of this layer abstract from common process modelling languages, and thus, it is expected, that the creation of process models even for inexperienced persons is simplified. Such approaches on top of business process models may pave the way for the detailed specification of requirements and elicitation of further design options and choices. Figure 6 shows the approach allowing an abstraction and concretion from depictive and descriptive models to Petri nets.

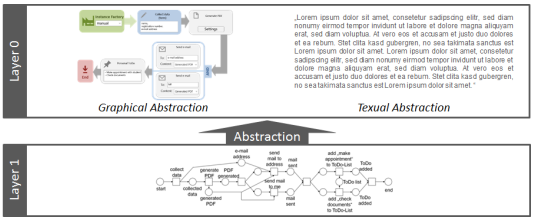



*Figure 6: Two modalities of the abstraction layer ("Layer 0")*

The design of business process models (for novices) can also be supported with a recommendation-based modelling support system (Koschmider et al. 2011). Such a system suggests how to finish appropriately a partially developed but incomplete business process model. The user can invoke the modelling support either via a query interface or s(he) can use a function automatically suggesting appropriate (Petri net based) process model fragments by unveiling the modelling intention of a user at process modelling time. Additionally, a social software-based extension was added to the recommendation-based modelling support in order to take into account a process builder's modelling context and the modelling history of a community of users (Koschmider et al. 2010). The system also suggests (Petri net based) process model parts to the user, that may help him achieving an individual modelling goal. Such features potentially improve the modelling process and, as such, the modelling outcome, that is, the quality of the process model. To support

location- and device-independent modelling a Petri net based tool has been suggested (Alpers and Hellfeld 2016). The pne.fzi.de tool supports continuous work on a business process model through intelligent synchronization mechanisms, even when changing devices. The list of Petri net-based extensions addressing the design phase can be completed with approaches addressing security and privacy issues or sustainability aspects in business process models (Betz et al. 2017). Petri net-based approaches for simulation were suggested in (Li and Oberweis 2009) and (Eichhorn et al. 2009). The approach proposed in (Eichhorn et al. 2009) tackles a 3D support for graphical business process simulation. A risk-aware simulation based on XML nets is presented in (Betz et al. 2011). In the future, process modelling languages should be capable of providing links to bind connected devices enabling sensing, (re-)acting, collecting and exchanging data via various communication networks including the Internet. So far, the design of business processes was dominated by a so-called Model-Enact paradigm, i. e., the process has been depicted as a (graphical) process model that afterwards could be executed by a Business Process Management System (BPMS) (Janiesch et al. 2017). The Internet-of-Things paradigm requires approaches that tackle the whole stack from sensor data to event data to process activities. Physical devices sense their environment and produce events that have to be correlated in order to distil complex events. Complex events are then used as input for process mining algorithms. Benefits and challenges of the integration of IoT into business process modelling are discussed in (Janiesch et al. 2017) and (Soffer et al. 2018). The list of future research fields can be complemented by the blockchain technology and the combination of augmented or virtual reality and business process modelling. 3D technologies open up new possibilities for modelling business processes. They provide higher plasticity and eliminate some deficits of conventional 2D process modelling such as the limitation of the amount of information to be integrated into a process model in an understandable way (Betz et al. 2008). The next section sketches approaches for the business process configuration phase relying on Petri nets.

## 3.2 Languages for the Configuration Phase

Technical realization of a process model is tackled by orchestration and choreography languages. WS-BPEL is the widely-used standard for this phase. It lays the foundation for a process engine automating the execution of business processes specified as (web) services. BPEL provides an XML notation and semantics for describing the behaviour of business processes. XML net based approaches were suggested since XML nets allow formal verification for the composition of services (Che et al. 2009). Particularly, with XML nets messages can be modelled and manipulated as place tokens for message passing, and the labels in arcs can be used to model constraints for web service discovery and selection. A further XML net based extension for web service composition was presented in (Koschmider and Mevius 2005) and (Lenz and Oberweis 2004). The paper (Koschmider and Mevius 2005) introduces so-called Web service nets allowing a process model driven deduction of BPEL. Web service nets describe control flows of web services and derive the description of web services including behavioural, functional, and interface-based information. For this purpose, four different types of flow structures are distinguished. Figure 7 shows the sequence, the alternative, parallelism and synchronization of activities. For instance, by the use of alternative, one branch of a set of choices can be selected.

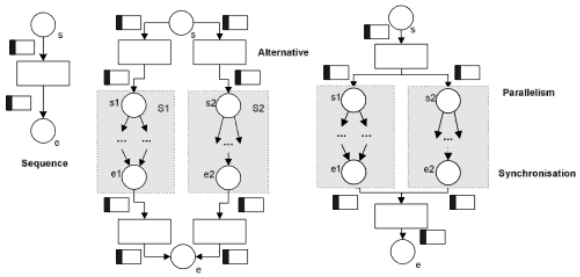



*Figure 7: Flow structures of Web service nets*

The execution of a business process might also depend on location constraints. A Petri net based method for defining location constraints in business process models was suggested in (Decker 2009), which is suitable for mobile business processes. Location constraints restrict the place where a process activity must be executed (positive restriction) or is not allowed to be executed (negative restriction). Figure 8 allows the execution of the activity "record order" only within the local instance "Berlin".

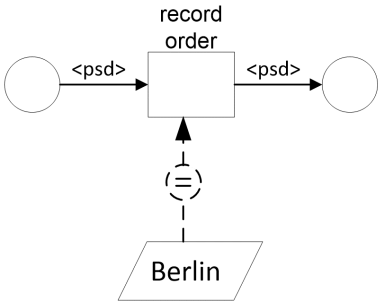



*Figure 8: Location constraints of mobile business processes*

One stream of future research for business process execution might stem from digital platforms, which require software architectures being capable of quickly adopting to changing conditions. In the future we will see research exploiting benefits of microservices for business process model execution. Microservice "*...is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms*" (Lewis and Fowler n.d.). In the context of Business Process Management (BPM) microservices could complement or even replace WS-BPEL for business process model execution. Contrary to WS-BPEL the characteristics of microservices allow executing highly distributed business process models without any centralized management. Particularly, process model activities or events could be specified as a microservice addressing its own technology stack. Challenges of a microservice-based business process execution are discussed in (Koschmider 2017b).

The next section sketches a Petri net-based approach for the enactment phase.

## 3.3 Languages for the Enactment Phase

One issue common to business process monitoring is the inability to link the business process schema to the information gathered during relevant phases (Mevius 2008). Therefore, (Lenz et al. 2005) presents an XML net based approach for process-oriented business performance management. In particular, the approach allows analysing potential exceptional states of performance indicators and matches counter measures in target-oriented simulation scenarios. Performance indicators are represented by XML schemas. Places that are typed by a performance indicator schema are interpreted as containers for the performance indicator value. The calculation of the performance indicator value is modelled by parts of an XML net and performance indicator violations can be described with alert or repair transitions. Figure 9 depicts the alert transition. An alert occurs if the ratio falls below a critical value.

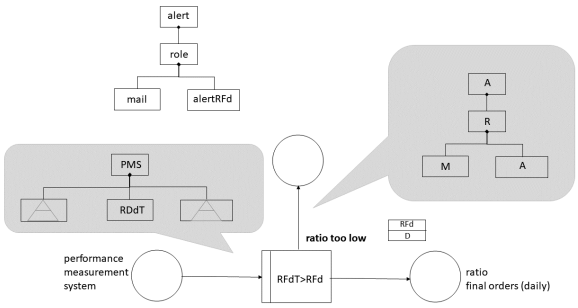



*Figure 9: Alert transition to describe performance indicator violations*

## 3.4 Languages for the Evaluation Phase

The evaluation phase is dominated by approaches for business activity monitoring (Friedenstab et al. 2012) and process mining (Aalst 2016).

The paper (Koschmider 2017a) describes a novel approach for clustering event traces by their

behavioural similarity. Existing process mining algorithms reportedly generate spaghetti models from event logs of flexible processes, which are largely incomprehensible. The technique presented in (Koschmider 2017a) can compare the control-flow of event traces rather than deriving a unique process model encompassing all traces. The comparison is related to time, duration and further exogenous factors such as temperature. The clustering technique is based on two algorithms, which classify behaviours and allow us to identify behavioural shifts. The improved classification of event traces makes the technique superior to existing approaches, allowing more efficient identification and analysis behavioural deviations (Koschmider 2017a). In the future process mining approaches are required allowing to combine heterogeneous sensor and event types.

## 4 Conclusion

In this paper we sketched business process modelling approaches relying on Petri nets. The approaches were classified following the business process life-cycle. Petri net based approaches for the modelling phase support modellers location- and device-independently in the construction of process models. The support ranges from tools that add an on top layer, to process models or automatically suggest process model parts for the completion of a process model editing activity. Approaches for the configuration phase use XML nets, a variant of high-level Petri nets, for web service-based compositions and executions of process models. Location constraints restricting the execution can also be defined. An extension of XML nets was also presented for the enactment phase allowing to describe performance indicator violations. Our contribution to the evaluation phase is a novel clustering technique for event traces.

In the future, process modelling languages should be capable of providing links to IoT devices and thus considering IoT parameters in process execution and evaluation (Janiesch et al. 2017).

## References

van der Aalst W. M. P. (2016) Process Mining - Data Science in Action, Second Edition. Springer

Alpers S., Hellfeld S. (2016) Werkzeug zur mobilen Modellierung von Geschäftsprozessen mittels Petri-Netzen. In: Betz S., Reimer U. (eds.) Modellierung 2016, 2.-4. März 2016, Karlsruhe - Workshopband. LNI Vol. 255. GI, pp. 147–152

Betz S., Eichhorn D., Hickl S., Klink S., Koschmider A., Li Y., Oberweis A., Trunko R. (2008) 3D Representation of Business Process Models. In: Loos P., Nüttgens M., Turowski K., Werth D. (eds.) Modellierung betrieblicher Informationssysteme. LNI Vol. 141. GI, pp. 73–87

Betz S., Fritsch A., Oberweis A. (2017) TracyML - A Modeling Language for Social Impacts of Product Life Cycles. In: Proceedings of the ER Forum 2017 and the ER 2017 Demo Track. CEUR Workshop Proceedings Vol. 1979. CEUR-WS.org, pp. 179–192

Betz S., Hickl S., Oberweis A. (2011) Risk-Aware Business Process Modeling and Simulation Using XML Nets. In: 13th IEEE Conference on Commerce and Enterprise Computing, CEC 2011. IEEE Computer Society, pp. 349–356

Che H., Li Y., Oberweis A., Stucky W. (2009) Web Service Composition Based on XML Nets. In: 42st Hawaii International International Conference on Systems Science (HICSS-42 2009)). IEEE Computer Society, pp. 1–10

Decker M. (2009) A Location-Aware Access Control Model for Mobile Workflow Systems. In: IJITWE 4(1), pp. 50–66

Desel J., Oberweis A., Reisig W., Rozenberg G. (1998) Petri Nets and Business Process Management. Dagstuhl seminars. Tech report.

Drescher A., Koschmider A., Oberweis A. (2017) Modellierung und Analyse von Geschäftsprozessen: Grundlagen und Übungsaufgaben mit Lösungen. DeGruyter Oldenbourg

Eichhorn D., Koschmider A., Li Y., Stürzel P., Oberweis A., Trunko R. (2009) 3D Support for Business Process Simulation. In: Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference, COMPSAC. IEEE Computer Society, pp. 73–80

Fahland D., Lübke D., Mendling J., Reijers H. A., Weber B., Weidlich M., Zugal S. (2009) Declarative versus Imperative Process Modeling Languages: The Issue of Understandability. In: 10th International Workshop on Enterprise, Business-Process and Information Systems Modeling, pp. 353–366

Figl K., Recker J. (2016) Exploring cognitive style and task-specific preferences for process representations. In: Requir. Eng. 21(1), pp. 63–85

Friedenstab J.-P., Janiesch C., Matzner M., Müller O. (2012) Extending BPMN for Business Activity Monitoring. In: 45th Hawaii International International Conference on Systems Science (HICSS-45), pp. 4158–4167

Janiesch C., Koschmider A., Mecella M., Weber B., Burattin A., Di Ciccio C., Gal A., Kannengiesser U., Mannhardt F., Mendling J., Oberweis A., Reichert M., Rinderle–Ma S., Song W., Su J., Torres V., Weidlich M., Weske M., Zhang L. (2017) The Internet-of-Things Meets Business Process Management: Mutual Benefits and Challenges. In: CoRR abs/1709.03628

Karhof A., Jannaber S., Riehle D. M., Thomas O., Delfmann P., Becker J. (2016) On the de-facto Standard of Event-driven Process Chains: Reviewing EPC Implementations in Process Modelling Tools. In: Modellierung 2016, pp. 77–92

Kindler E. (2006) On the semantics of EPCs: Resolving the vicious circle. In: Data Knowl. Eng. 56(1), pp. 23–40

Koschmider A. (2017a) Clustering Event Traces by Behavioral Similarity. In: ER 2017 Workshops. Lecture Notes in Computer Science Vol. 10651. Springer, pp. 36–42

Koschmider A. (2017b) Microservices-based Business Process Model Execution. In: 8th International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA), pp. 158–161

Koschmider A., Caporale T., Fellmann M., Lehner J., Oberweis A. (2015) Business Process Modeling Support by Depictive and Descriptive Diagrams. In: 6th International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA), pp. 31–44

Koschmider A., Hornung T., Oberweis A. (2011) Recommendation-based editor for business process modeling. In: Data Knowl. Eng. 70(6), pp. 483–503

Koschmider A., von Mevius M. (2005) A Petri Net Based Approach for Process Model Driven Deduction of BPEL Code. In: OTM 2005 Workshops. Lecture Notes in Computer Science Vol. 3762. Springer, pp. 495–505

Koschmider A., Song M., Reijers H. A. (2010) Social software for business process modeling. In: JIT 25(3), pp. 308–322

Lenz K., von Mevius M., Oberweis A. (2005) Process-Oriented Business Performance Management with Petri Nets. In: 2005 IEEE International Conference on e-Technology, e-Commerce, and e-Services. IEEE Computer Society, pp. 89–92

Lenz K., Oberweis A. (2003) Inter-organizational Business Process Management with XML Nets. In: Petri Net Technology for Communication-Based Systems - Advances in Petri Nets. Lecture Notes in Computer Science Vol. 2472. Springer, pp. 243–263

Lenz K., Oberweis A. (2004) Workflow Services: A Petri Net-Based Approach to Web Services. In: International Symposium on Leveraging Applications of Formal Methods, ISoLA 2004. Technical Report Vol. TR-2004-6, pp. 35–41

Lewis J., Fowler M. (n.d.) Microservices: a definition of this new architectural term.. https://martinfowler.com/articles/microservices.html

Li Y., Oberweis A. (2009) A Petri Net-Based Software Process Model for Developing Process-Oriented Information Systems. In: Proceedings of Information Systems Development ISD. Springer, pp. 27–39

Mendling J., van der Aalst W. M. P. (2006) Towards EPC Semantics based on State and Context. In: 5. Workshop der Gesellschaft für Informatik e.V. (GI) und Treffen ihres Arbeitskreises "Geschäftsprozessmanagement mit Ereignisgesteuerten Prozessketten (WI-EPK)", pp. 25–48

von Mevius M. (2008) A novel modeling language for tool-based business process engineering. In: Proceedings of the 2008 ACM Symposium on Applied Computing (SAC). ACM, pp. 590–591

Nüttgens M., Rump F. J. (2002) Syntax und Semantik Ereignisgesteuerter Prozessketten (EPK). In: Prozessorientierte Methoden und Werkzeuge für die Entwicklung von Informationssystemen. LNI Vol. 21. GI, pp. 64–77

Oberweis A. (1996) Modellierung und Ausführung von Workflows mit Petri-Netzen. Teubner-Reihe Wirtschaftsinformatik. Teubner

Petri C. A. (1962) Fundamentals of a Theory of Asynchronous Information Flow. PhD thesis, pp. 386–390

Radloff M., Schultz M., Nüttgens M. (2015) Extending different Business Process Modeling Languages with Domain Specific Concepts: The Case of Internal Controls in EPC and BPMN. In: Proceedings of the 6th Int. Workshop on Enterprise Modelling and Information Systems Architectures (EMISA), pp. 45–58

Reisig W. (2013) Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies. Springer

Soffer P., Hinze A., Koschmider A., Ziekow H., Ciccio C. D., Koldehofe B., Kopp O., Jacobsen A., Sürmeli J., Song W. (2018) From event streams to process models and back: Challenges and opportunities. In: Information Systems to appear

Weske M. (2012) Business Process Management - Concepts, Languages, Architectures, 2nd Edition. Springer

# Hierarchical robustness model for business processes

Natalja Kleiner[a], Peter C. Lockemann[*,a]

[a] FZI Forschungszentrum Informatik, Karlsruhe, Germany

Abstract. *Most business processes today can be easily modelled and controlled by advanced business process management systems. When it comes to processes, that are driven by outside events and require fast reactions to contingencies in order to stabilize them, e. g., transport or production processes, business process management systems often seem to reach their limits. In this paper we introduce a robustness model which is based on a business process model of an undisturbed transport process and extends it by a generic contingency detection model. We employ a hierarchical organization for deriving corrective actions with least possible modifications to the original transport process.*

Keywords. Logistics • Failure Management • Failure Model • Complex Event Processing

## 1 Motivation

Business processes are usually modelled by workflow or business process techniques. Special software – workflow or business process management systems – is employed to drive the process across the actions in the model. The underlying assumption is that most of the actions take place in the information processing world and can easily be controlled by the management system.

Processes in logistics, e. g., in production or transport, follow a sequence of steps, sometimes with some alternatives provided. Hence one would expect that these processes could easily be modelled by workflow or business process techniques. Contrary to the assumptions above, however, the processes are driven by outside events in the real world and not a piece of software. Hence, a process management system by itself would not make sense. Further, the major actions are real-world activities such as loadings, transports, deliveries. Just a few actions reflect information processing activities, mostly in a supporting role, e. g., by registering the state of the process or by automating accompanying paperwork.

* Corresponding author.
E-mail. lockemann@fzi.de

We claim in this paper that business process models could still form an important base in logistics, though in a more special manner. Numerous contingencies may arise along a transport process. Suppose that a business process model reflects the regular, undisturbed transport process, then what we plan to achieve is to use the model as a framework for deriving corrective or re-planning actions in a systematic fashion.

The article is organized as follows. In section 2 we briefly touch on related work. In section 3 we give an example scenario. Section 4 introduces a generic model for the disturbances along the transport process. Section 5 builds on the model and develops a system that drives the necessary corrections to the process. Section 6 concludes the paper.

## 2 Related work

Systems, which withstand disturbances, are called *robust*. To be more precise, Wikipedia defines robustness 'as the ability of a system to resist change without adapting its initial stable configuration', a definition that is only helpful if one specifies what is meant by 'change' or 'stable'. Also, the definition has a static flavour. A more process-oriented, dynamic view describes robustness as

the capability of a process to function reliably even under unfavourable conditions (Vogel et al. 2009). Again, whether a process is considered robust depends on the pertinent definitions of 'reliable' and 'unfavourable'.

In their general classification of system dependability (or reliability) aspects, Laprie et al. (1992) distinguish two ways of coping with failures in dependable systems: fault prevention and fault tolerance. Fault prevention is concerned with how to prevent fault occurrence, and is, to a large degree, a design issue and requires design rules which help to avoid introducing failures in a system. Fault tolerance deals with how to provide a service complying with the specification in spite of faults. Since transport logistics is driven by external forces, extraneous faults and failures seem unavoidable. Hence fault tolerance is the dependability issue. One of the first authors to consider robustness in a logistics context have been Wieland and Wallenburg (2012).

Planning for robustness relies on a list of expected failures and describes alternatives to be taken when a particular failure arises. Hagen and Alonso (2000) suggest that due to the control system complexity one should separate failure handling aspects from the normal flow of control. They demonstrate the principle with an approach from a transaction perspective and propose atomicity and exception handling as the two fundamental techniques to deal with fault tolerance. In case of a failure, an application or parts of it are rolled back to a previous consistent state (backward recovery). From this state, the computation continue by following alternative or compensation execution paths (forward recovery).

Such an approach makes sense if most of the actions are confined to information processing. In transport logistics, by the time a failure has been detected, the process has left too many irreversible traces in the real world to have a chance to return to a previous state. Hence, forward recovery is the only way to proceed. Standard approaches in workflows are either to dynamically modify the workflow at the point of failure, or to provide a set of mini-workflows to execute in place of the failed action. For an example, see, e. g., Lanz et al. (2010). As part of the Workflow Pattern Initiative, Russell et al. (2006) group unanticipated events into classes which are related by similarities in terms of conditions under which they may arise. Based on these, they develop patterns, i. e. generic recurring constructs, to incorporate in a workflow. Cognini et al. (2016) introduce richer sets of modelling constructs in the form of business process fragments and variants that include a wide repository of constraints.

As we shall demonstrate below, failures in transport logistic are of varying severity, and the scope of their effect across a network may differ considerably. Current solutions do not seem to account for those variations, at least not in a systematic fashion. We present a novel approach that organizes failures into a hierarchy, where those on higher levels have a wider effect than those on lower levels. We associate with each failue a compensating action with a concomitent reach of the effect. We escalate the actions up to the level where a suitable action can be found.

## 3 An example: Open logistics networks

The overall objective of transport logistics is to provide the desired goods in the correct volume at the right time and right place. Stakeholders are the suppliers, customers and transport carriers are the intermediaries. Since stakeholders of all three kinds interact in numerous ways, they form a network. An open network is one where stakeholders may freely enter or disconnect, and are free to enter temporary collaborations, e. g., to share transfer orders, improve the utilization of load capacities, or to help out in damaging situations. Particular challenges arise in a clocked network with carriers, which must follow a rigid timetable, e. g., railway companies. Figure 1 illustrates the principle.

Below on the left, goods are collected from three suppliers via separate trucks and the consolidated in a first transition hub (so-called hub-and-spoke principle), whereas on the right a single truck collects the wares from two suppliers and delivers

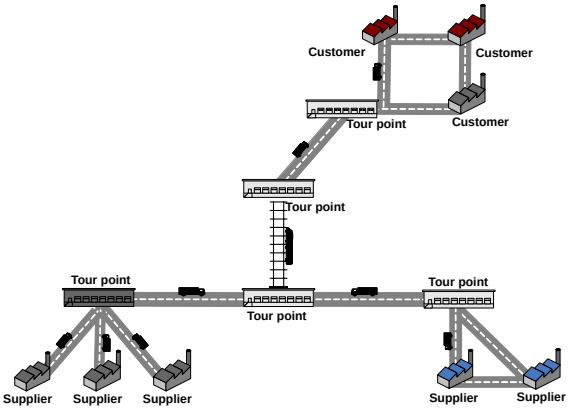



*Figure 1: Example of a logistics network*

them to a second hub (a so-called round trip). Both hubs will send trucks (perhaps after consolidating the previous loads with wares from further sources) to a so-called railport. Note that nothing has been said on whether the loads belong to one or more orders or whether they go to one or more customers. The rail hub by necessity consolidates a large number of orders into the load of a complete freight train which leaves according to a predetermined schedule. Likewise, at the other end a corresponding distribution over several legs will take place. From now on we will refer to points of loading or unloading and hubs collectively as tour points and to the transport between two tour points by a single vehicle as a tour.

The entire transport chain can be modelled by any suitable formalism, e. g., BPM or UML. In the remainder we abstract from any particular formalism.

## 4 Basic elements of failure management

### 4.1 Robustnesss

Whether a logistics network is robust or not is in the eye of the beholder. Since there are many stakeholders, one can expect that each of them holds an individual view on robustness. If we take the network as a whole, then, according to section 3, we should consider a logistics network robust if an order placed by a customer with a supplier is delivered at the specified place and time in correct composition and volume (see Magnus and Thonemann 2007), and that this should hold for every customer-supplier relationship within the transport system.

During transport, deviations from schedules or routes are the norm. Fortunately, most of them remain 'under the radar', i. e., are not noticed at all or considered inconsequential. For a deviation, that can no longer be safely ignored, but should at least be tracked because it could endanger the robustness of the transportation process, we use the term *exception*. Not every exception will have repercussions on the normal course of events and actions. Those that do will be termed *contingencies* because they require some sort of counteraction. *Failure* serves as a generic term to cover both, exceptions and contingencies.

### 4.2 Exceptions and Contingencies

Transport systems are more or less continuously monitored by collecting various measurement data from outside, using technical devices like RFID scans, GPS tracking, thermometers, pressure meters, and, sometimes, human observations. As long as these do not raise an alarm, a disruption and the resulting deviation remain unobserved. When they do, a deviation may qualify as an exception or a contingency.

On the physical level the exceptions and contingencies have to do with disruptions, which generally result in delays. Take traffic jams, detours, driver's indispositions, truck or train breakdowns, missing, incomplete, defect or incorrect shipments at a supplier, non-available ramps or storage areas at hubs, unpreparedness at the customer site, incorrect transport documents, vehicle replacements of the wrong type, to name a few. Delays may result in failures of much farther reach. Take a delay due to vehicle breakdown. Suppose a replacement vehicle can be found. This will affect the entire truck fleet of a transport company or even a second company. And even after one has been found, it may turn out that it has insufficient load capacity to pick up all the goods along the tour it was scheduled to do. In the worst case the transport may miss a scheduled train and wait for the next train, causing delivery of the order

at the customer site to be late by many hours or even a day. Disruptions may not only originate on the physical level. Consider emergency repairs in a vehicle fleet nullifying a tour plan, or the cancellation of parts of an order or an increase of volume of an order.

We conclude that although deviations may occur on a local level, their effects may reach much farther. A careful analysis of logistics networks shows that one can distinguish five spheres of influence that form a hierarchy (see figure 2). As mentioned above, some deviations may even originate on higher levels.

### 4.3 Buffers

A time-honored approach to robustness is to build some slack into the business process. Basically one adds some reserve capacity to the resources employed in the process (Bretzke 2010). For example, to overcome delays one allows more time than absolutely needed for the tour, or earlier arrivals than in the exact plan. Likewise, one may provide larger load capacity than minimally needed. Of course, this comes at a price because additional or larger trucks must be kept in reserve. We refer to the spare resources as *buffers*.

In designing failure systems one has first do define what one considers resources. Are they exclusively of a physical nature such as trucks or personnel, or also conceptual such as tour times or tour point sequences? Given these one can then assign buffers to them. Next, one can estimate how far their influence may reach, and arrange them within the hierarchy of spheres. For example, tour time buffers or load capacity buffers belong to the tour level because they can be used to alleviate disruptions of a single tour. Further examples for this level are alternative routes or alterations in the sequence of tour points. Typical examples on the transport level, where an entire transport from a supplier to a customer is considered, are excess loading personnel, several scheduled trains or unused warehouse capacities in the hubs. On the order level we observe buffers like delivery time or order splits. Typical for the fleet level are reserve trucks, and for the top, network level alternative suppliers, customers or transport carriers. Figure 2 illustrates the hierarchical arrangement of buffers.

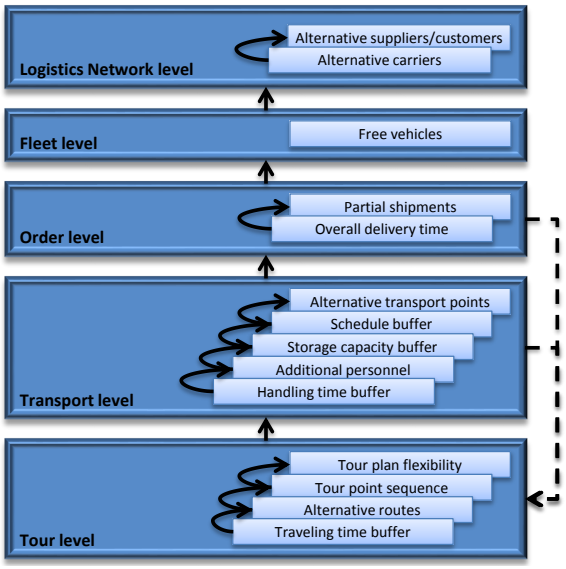



*Figure 2: Buffer hierarchy*

### 4.4 Tolerance intervals

Even though the business plan merely seems to describe the logistics process rather than drive it, it evidently plays an active role in case of failures. Certain buffers are statically specified as part of the plan, but are then consumed whenever spare resources are used for overcoming an exception. By drawing on the buffers and dynamically adjusting them, the business plan seems to occupy the driver's seat once an exception occurs.

For example, compensating for a delay may consume half of a delay buffer at a given tour point. Since a delay buffer can be associated with, an entire round trip the other half is still available for further delays along the run. Hence, the resource represented by a buffer seems to shrink dynamically. We refer to the dynamic counterpart of a buffer as a *tolerance interval*.

## 5 Failure handling

### 5.1 Deviation detection

As mentioned, the external system raises an alarm if a worthwhile disruption has been observed. This

alarm is usually referred to as an *event*. In fact, many events caused by observations may only become meaningful if they are considered in a larger context. Just consider temperature measurements, where a dangerous situation is recognized early enough after successive values show a growing tendency. Or suppose that GPS tracking indicates that a truck is behind its expected position, and the traffic situation indicates that late arrival seems unavoidable.

Consequently, that makes it less than straightforward to detect an exception. On arrival of an event, the event must be checked of whether it may become part of a complex event, conditions must be checked, the data accompanying the event must be compared with a specified target value, and finally a decision be taken whether the event qualifies as a deviation. Deviation detection, therefore, must be captured in the form of a set of rules, e. g., ECA (event-condition-action) rules.

## 5.2 From deviation to contingency

Once a deviation has been detected one must determine which resources are affected, and hence which sphere should be examined. Since we know the current position in the workflow we can infer the resources. Take late arrival at a tour point. Then trucks, personnel and loading ramps are candidates. Associated with each resource are certain quality characteristics, e. g., truck schedule, load capacity, ramp assignments. These define the buffers to be inspected. Note that more than one resource may be affected, e. g., given a delay both the schedule of a truck and the work schedule of its driver should be examined.

Next, we must examine the candidate buffers and their tolerance intervals on the given hierarchy level. This should be done in a certain order. The arrows in figure 2 give an example. For each interval we subtract the deviation associated with the failure from the current value. If the interval remains above zero then the buffer could accommodate the deviation and therefore the deviation can be classified as an exception. If necessary, we go on and inspect the next buffer. For example, if a truck arrives late but within tolerance, and the driver can take a rest within tolerance all intervals remain above zero and the effect remains strictly local. If not, we may try to alter the sequence of tour points next. If this keeps us within tolerance – now for the entire tour – we still keep the effect within the tour level. However, because the correction now involves re-planning the tour, the deviation has morphed into a contingency. Re-planning is usually complicated. In finding a sequence, the tolerance intervals of the other tour points must be observed. In case of finding an alternative route one must employ routing algorithms. Re-planning algorithms are again associated with the buffers.

If the failure remains (locally) an exception, then it will not by itself affect the buffers of the successive transport points, because each buffer has been designed independent of all the other buffers. However, on arrival at the next point the intervals must be properly adjusted.

Whenever the intervals within a given level have been exhausted, i. e., at least one tolerance interval has fallen below zero we have a contingency that cannot be dealt with on the current level. Suppose that a truck has been delayed for too long to load wares at a transport point within the tolerance interval. One remedy could be to start a second truck on time to pick up the wares. However, this affects resources beyond the tour level. Figure 3 illustrates the procedure.

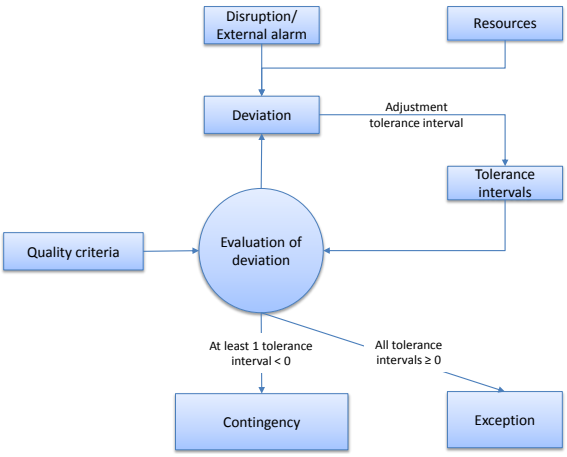



*Figure 3: Failure handling on single hierarchy level*

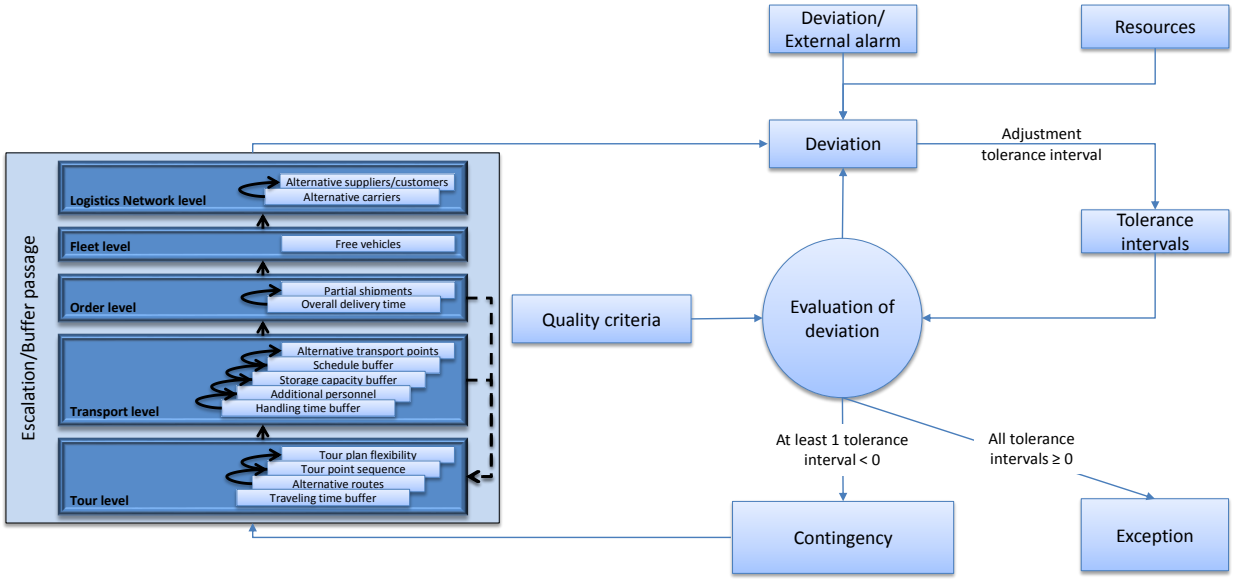



*Figure 4: Escalated failure handling*

### 5.3 Escalation

In the previous example the contingency had a reach beyond the current level: The suggested remedy cannot be dealt with locally but affects the carrier's fleet. A buffer reflecting spare trucks would have to be provided on a higher level, in this case the fleet level. This gives us a handle on how to treat contingencies: Escalate the contingency to a higher level and hope that the contingency is contained on that level. To give another example, in a tour level contingency one may examine the entire transport chain for additional slack due to delay. The transport level is aware of all the tour points along the transport chain and thus may inspect each of them whether they still could tolerate the original delay. It may discover that the railport offers, with its temporal and volume buffers, the possibility to take the next train without a violation of tolerance intervals further along the transport chain.

The basic idea of our approach is to try to resolve the contingency on higher levels of the hierarchy. Basically, the contingency is propagated to the next higher level (failure escalation). In principle, since each level in the hierarchy has its own set of resources with their commensurate buffers, the procedure of section 5.2 applies separately to each level in the hierarchy. Figure 4 illustrates the propagation within each level and the escalation across levels. Note that exceptions may originate on any level. For example, a customer may change an order while the underlying tour is already on its way, or a supplier may have to split an order into several parts.

Escalation is more complicated than propagation within a single level. Many or all tours within the transport chain must be individually replanned with the target utilizations and their buffers adjusted. Thus, escalation means change propagations up and down the hierarchy, and this perhaps several times (figure 4).

## 6 Conclusions

In general, business plans serve two purposes. First, they are a planning tool and as such document the business intentions. Second, by embedding them within a business process management system they become a vehicle to drive the business process. If large portions of the business process take place in the physical world, the second reason does not apply. Even then, as we have shown, a business plan may assume an active role, albeit as

the means to guarantee the robustness of a business process. To do so, one will have to embed the business plan into a failure management system, and augment the plan by resource buffers and associated re-planning algorithms.

Using the real-world example of a logistics network we were also able to show that if robustness is a matter of many resources and stakeholders resulting in spheres of influence of different width, a hierarchical approach eases the design of robustness and adds transparency.

We believe the approach is sufficiently generic to be applicable to other application domains, e. g., to production scenarios. However, further research is still needed to confirm the assumption.

*The work reported is part of a dissertation by Natalja Kleiner. The paper is a bit of a historical reminiscence – Natalja is the second author's final student while H.C. Mayr was one of his early fellow researchers.*

## References

Bretzke W. (2010) Logistic Networks (in German). Springer Berlin Heidelberg

Cognini R., Corradini F., Polini A., Re B. (2016) Business Process Feature Model: An Approach to Deal with Variability of Business Processes In: Domain-Specific Conceptual Modeling: Concepts, Methods and Tools Karagiannis D., Mayr H. C., Mylopoulos J. (eds.) Springer International Publishing, pp. 171–194

Hagen C., Alonso G. (2000) Exception handling in workflow management systems. In: IEEE Transactions on Software Engineering 26(10), pp. 943–958

Lanz A., Reichert M., Dadam P. (2010) Robust and Flexible Error Handling in the AristaFlow BPM Suite In: Information Systems Evolution: CAiSE Forum 2010, Hammamet, Tunisia, June 7-9, 2010, Selected Extended Papers Soffer P., Proper E. (eds.) Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 174–189

Laprie J. C., Avizienis A., Kopetz H. (eds.) Dependability: Basic Concepts and Terminology. Springer-Verlag New York, Inc., Secaucus, NJ, USA

Magnus K., Thonemann P. (2007) Successful Supply Chain Cooperation Between Retailers and Consumer Goods Manufacturers: An Empirical Study of the Retailers Perspective (in German). Gabler Edition Wissenschaft. Deutscher Universitätsverlag

Russell N., van der Aalst W., ter Hofstede A. (2006) Exception Handling Patterns in Process-Aware Information Systems. BPM-06-04. BPM Center. http://bpmcenter.org/wp-content/uploads/reports/2006/BPM-06-04.pdf

Vogel O., Arnold I., Chughtai A., Ihler E., Kehrer T., Mehlig U., Zdun U. (2009) Software Architecture - Basics, Concepts, Applications (in German). Spektrum Akademischer Verlag

Wieland A., Wallenburg C. M. (2012) Dealing with supply chain risks: Linking risk management practices and strategies to performance. In: International Journal of Physical Distribution & Logistics Management 42(10), pp. 887–905

# A Short Comparison of Business Process Modelling Methods under the Perspective of Structure and Behaviour

Elmar J. Sinz*,a

[a] Faculty of Information Systems and Applied Computer Sciences, University of Bamberg, Germany

Abstract. *EPC, BPMN, SOM and petri nets are methods to business process modelling which look quite different at the first glance. Considering the two main characteristics of a system, structure and behaviour, this short article shows two things: (1) in all methods the behaviour model can be regarded as a petri net enriched with certain semantics, (2) the structure model is missing in all methods besides SOM, thus wasting a lot of semantics.*

Keywords. Business Process Modelling • EPC • BPMN • SOM • Petri Net

## 1 A system view on business processes

A business process can be understood as (1) collection of activities, separated by means of common attributes, (2) event-driven flow of these activities, (3) adoption of inputs and generation of outputs having a value for the consumers, and (4) assignment and utilization of some resources (Ferstl and Sinz 1993; Vossen and Becker 1996). From this follows that a business process can be understood as a system, consisting of components and relationships.

What is a business process under the perspective of structure and behaviour? Structure and behaviour are the main characteristics of a system. Compared with a transportation system, structure is the network of roads connecting the components; behaviour is the traffic on it. The structure of a system determines the scope of its behaviour; a certain behaviour of a system is only possible, if the structure supports it.

As an example, a business process involving a company and a customer is used. The business process starts with product information sent from the company to the customer. In case it shows what the customer wants, the customer places an order which is returned to the company. Processing the order, the company submits a shipping order to its store, which releases the shipping to the customer. An internal shipping report finishes the business process.

In the following, it is shown that EPC, BPMN and SOM model the behaviour of a business process based on petri nets. Despite looking quite different, the methods can be led back to the same notation. On the other hand, only SOM looks at the structure of a business process.

## 2 Petri Nets as a Basic Method for Business Process Modelling

Petri nets (Reisig 2010) are a basic notation for modelling information flows of systems. They consist of two types of components, places and transitions. Transitions can have input places as well as output places. In the simplest case, a transition can fire, if all its input places are at least marked with one token, causing all its output places to get an additional token. Thus, a petri net models only the behaviour of a system, its structure is ignored. The components of the systems, e. g. company and customer, cannot be shown.

Of course there are different ways to model a business process as a petri net, depending on

* Corresponding author.
E-mail. elmar.sinz@uni-bamberg.de

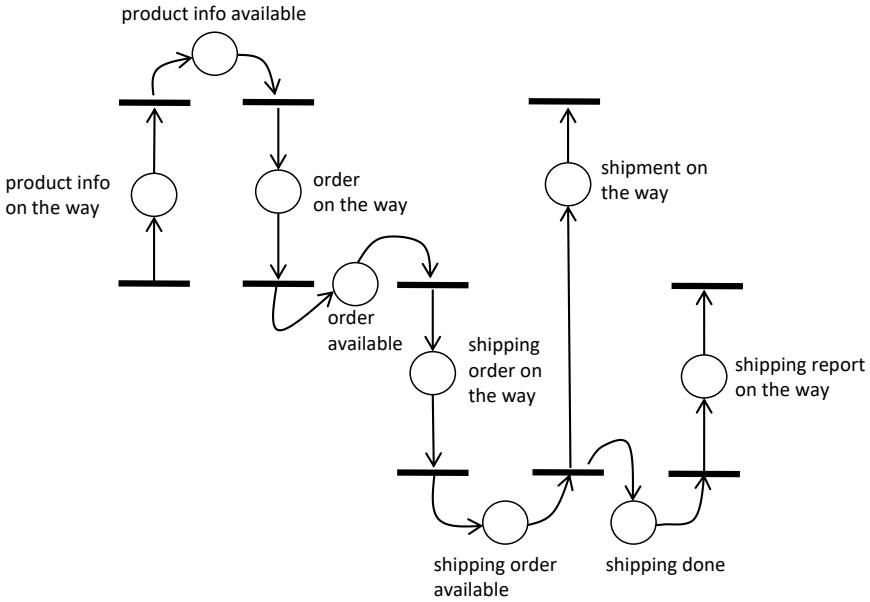



*Figure 1: The Sample Business Process as a Petri Net*

the goals of the model. For example, the transfer of information or goods can be modelled using separate places, or can be suppressed. In the following, for reasons of comparability of the different business process modelling methods, the transfer is modelled.

Figure 1 shows the resulting business process as a petri net. It starts on the left side with the product information, followed by an order if the product information conforms to the requirements. The order is succeeded by a shipping order, causing the shipment and the generation of a shipment report.

## 3 Event-Driven Process Chains (EPC) as a Method for Business Process Modelling

Event-driven process chains (EPC) (Nüttgens 2017) are a well-known method for business process modelling proposed in the ARIS approach (Scheer 1998). Figure 2 shows the business process of Figure 1 represented as EPC. To facilitate the comparison of the two methods, conforming components are arranged similarly. The concepts are bridged as follows:

- Events (represented as hexagons) correspond one-to-one to the places of petri nets.
- Transitions are replaced by functions. While a transition is not time-consuming, a function may be.
- Connectors (not used in Figure 2) are represented by circles labelled with the Boolean operators AND, OR and XOR. They can be combined and used to specify pre- and post-conditions of functions. Compared to petri nets, connectors are an additional feature, leading to an extension of semantics. In a petri net, the equivalent of a connector has to be modelled by additional transitions and places, considering that a transition can fire if all preceding places are at least marked once (AND) or all preceding transitions feed a common place which is the single input for the particular transition (OR).

A remarkable difference between a petri net and an EPC is that an EPC always starts and ends with an event (Scheer and Thomas 2005). Thus, in the current example three additional events have to be included (in Figure 2 grey shaded).

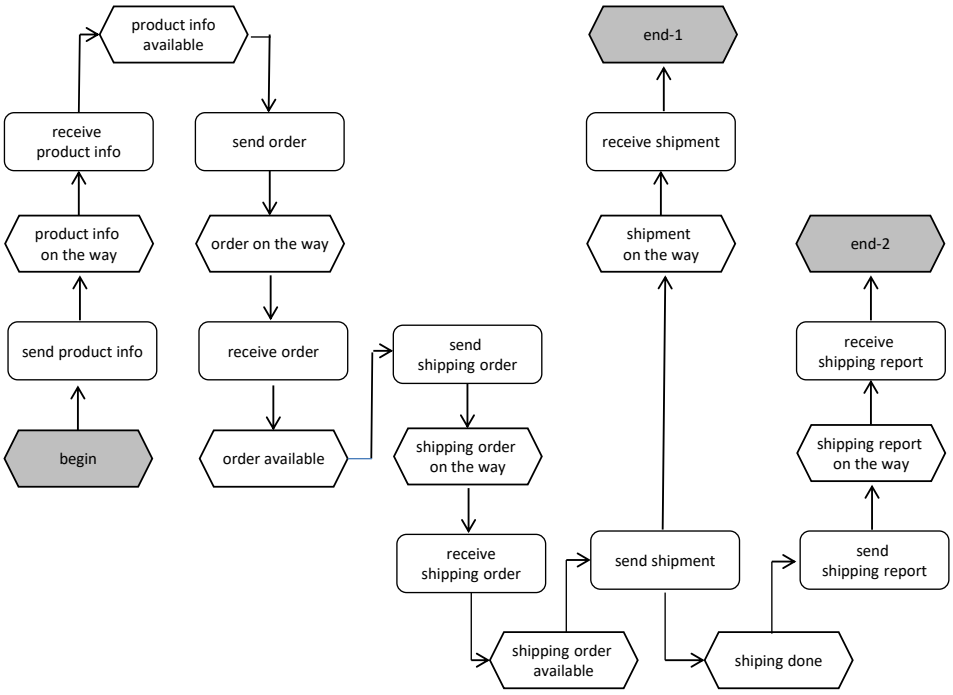



*Figure 2: The Sample Business Process as an Event-Driven Process Chain*

Functions can be related with e.g. information objects and organizational units. These are referencing items only, they do not constitute an own model within the context of an EPC.

## 4 Business Process Model and Notation (BPMN) as a Method for Workflow Modelling

Business Process Model and Notation (BPMN) is a popular method for workflow modelling (OMG 2017; Weske 2012). The term workflow instead of business process means that a workflow specifies the activities and relations between activities while executing one or more business tasks (Pütz and Sinz 2010). By contrast, a business process describes business tasks and event relations between tasks. It is strictly goal-oriented. Workflow models focus as well as business process models on the behaviour of a system.

The corresponding concepts between a petri net and a BPMN schema are bridged as follows (Figure 3):

- Transitions correspond to activities.
- Places occur as start events, as end events, or as events connecting activities. The latter can be omitted due to the fact, that a BPMN schema models a workflow. Here, events inside the execution of a business task are not in the foreground.
- Gateways (not used in Figure 3) allow e.g. the parallel split or the merging of flows.

The most interesting concepts are pools. Pools are participants, shown as rectangles which surround activities. The rectangles are labelled with the names of the participants, who execute the corresponding activities. Inside a pool, activities are related by sequence flows (solid lines); between pools there are message flows (dashed lines).

Another semantic detail can be traced back to the fact that a pool is a participant. Compared to a petri net, the control flow of activities executed by one participant is continuous. For example, the sequence flow between "send product info" and "receive order" cannot be found in the petri net (Figure 1) and by the way neither in the EPC (Figure 2).

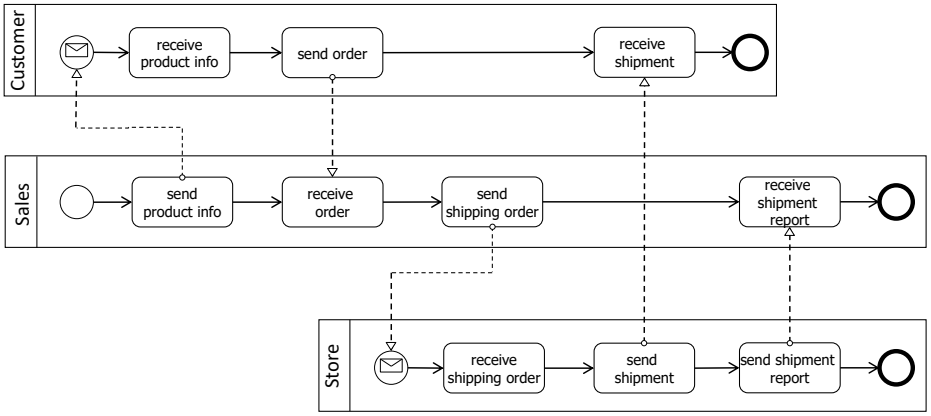



*Figure 3: The Sample Workflow as a BPMN Schema*

## 5 Semantic Object Model (SOM) as a Method for Business Process Modelling

In the SOM method (Ferstl and Sinz 1995; Ferstl and Sinz 2005; Ferstl and Sinz 2013, p. 194) the behaviour of a business process is modelled by a task-event schema. The following issues bridge between the concepts of a petri net and a task-event schema (Figure 4):

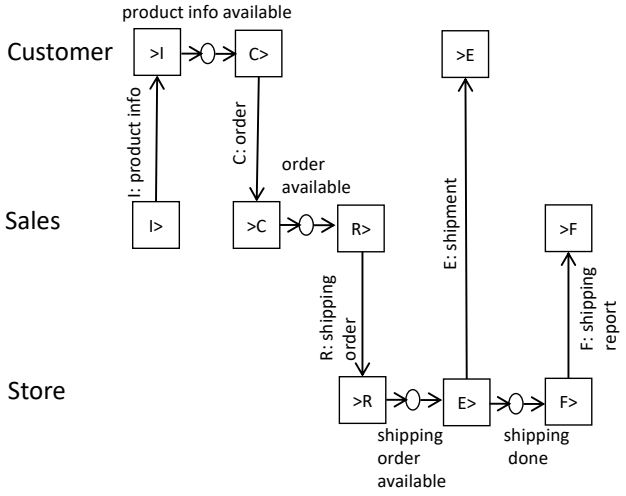



*Figure 4: The Sample Business Process as a SOM Task-Event Schema (Behaviour)*

- Tasks: Transitions are replaced by tasks. A task is a goal-oriented operation on a task object, released by and producing events.
- Object-orientation: Tasks are combined in an object-oriented way. All tasks operating on the same task object form an object. In Figure 4 e. g. the object customer contains the tasks >I (receive a product info), C> (send an order), and >E (receive a shipment).
- Places correspond to internal events of an object, i.e. an event connecting two tasks of one object.
- Transactions: An event from an object to another is represented as a transaction. A transaction causes a synchronized execution of the two tasks, e.g. C> (send an order) and >C (receive an order) must be completed in one transaction. C> as well as >C cannot terminate separately. Therefore, the event of a transaction is not displayed in the task-event schema.
- Pre- and post-conditions: Tasks can be complemented by pre- and post-conditions. These are Boolean expressions, e.g. if shipping is done only once in the afternoon, task E> could have the pre-condition "shipping order available AND time = 5 p.m.".
- Colored petri nets: the tokens of an event can be distinguished and therefore assigned to an instance of a task operation.

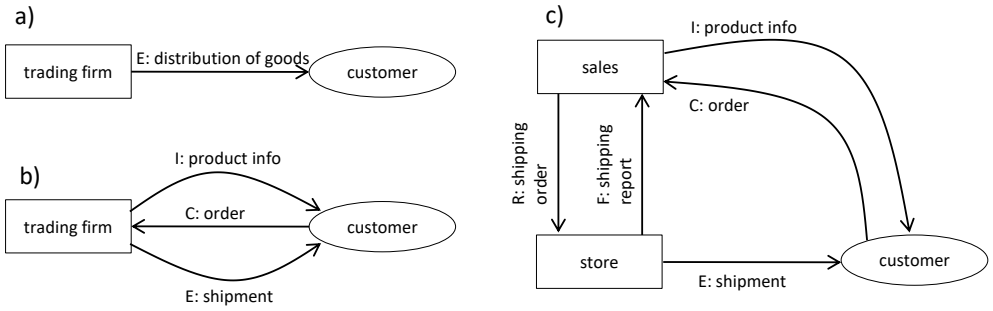



*Figure 5: The Sample Business Process as alternate SOM Interaction Schemas (Structure)*

Besides the task-event schema as the behaviour view on a business process, SOM provides an interaction schema as the structure view. Structure and behaviour view are adjusted, but an advantageous modelling always starts with the structure view. The structure view shows the decomposition of a system or a business process respectively, thus revealing sub-transactions and sub-objects (Figure 5).

A SOM interaction schema always starts with the most aggregated system view on the universe of discourse (trading firm) and its environment (customer). The object denoting the universe of discourse is connected with each environment object by one transaction (distribution of goods) (Fig 5a). Now the transaction(s) or the universe of discourse have to be further decomposed. It's a good idea to continue with the transaction, which is deconstructed according to the rule

$$T(O, O') \ ::=$$

$$[[\ T_i(O, O')\ seq]\ T_c(O', O)\ seq\ ]\ T_e(O, O').$$

This means: replace the transaction on the left side ($T(O, O')$) by an initiating transaction ($T_i(O, O')$) sequentially followed by a contracting transaction ($T_c(O', O)$) sequentially followed by an enforcing transaction ($T_e$) (Fig. 5b). Initiating transaction as well as initiating and contraction transaction can be omitted. The rule is called the negotiation principle and is one of the two fundamental coordination principles the SOM model supports. The other coordination principle is the feedback control principle. It is given by the rule

$$O \ ::= \ \{\ O',\ O'',\ T_r(O', O''),\ [\ T_f(O'', O')\ ]\ \}.$$

An object $O$ is replaced by the set of objects $O'$ and $O''$, a control transaction $T_r(O', O'')$ from $O'$ to $O''$ and a feedback transaction $T_f(O'', O')$ from $O''$ to $O'$. The latter can be omitted if there is only a controlled system. As shown in Figure 5c, the object trading firm is decomposed into the sub-objects sales and store as well as the control transaction shipping order and the feedback transaction shipping report. The sub-transactions from the first decomposition are linked to the new sub-objects.

Given an appropriate software tool (Ferstl et al. 2016), one can slide up and down the object decomposition as well as the transaction decomposition, each level showing a consistent decomposition of the system and associated with a corresponding task-event schema.

## 6 Conclusion

As pointed out, EPC, BPMN and the task-event schema of SOM can be explained in terms of petri nets. The modelling methods amend the petri net semantics differently to specify the behaviour of a business process or a workflow respectively. For example, EPC denote every event and are easy to read, BPMN show participants and their communication using message flows, and SOM points out the synchronous execution of a transaction between different objects.

Only SOM has a structure model, having a lot of benefits. The structure shows the decomposition of a system revealing sub-objects and sub-transactions. The reason for having a structure model is the object-orientation of SOM. The combination of "objects having tasks" and "two tasks of different objects are driving a transaction" is the prerequisite for the decomposition of a model.

The opportunity to zoom in and out the system and having a consistent model on each level adds a "third dimension" to business process modelling. On each level of aggregation a behaviour model can be assigned. The structure model is among others the platform for model driven architecture (e. g. Pütz and Sinz 2010).

BPMN could be amended with a structure model when giving up the semantics of "a pool is a participant" and replacing it by "a pool is an object". Without investigating all the details, this would be a great step forward.

## References

Ferstl O. K., Sinz E. J. (1993) Geschäftsprozeßmodellierung. In: WIRTSCHAFTSINFORMATIK 35(6), pp. 589–592

Ferstl O. K., Sinz E. J. (1995) Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen. In: WIRTSCHAFTSINFORMATIK 37(3), pp. 209–220

Ferstl O. K., Sinz E. J. (2005) Modeling of Business Systems Using SOM. In: Bernus P., Mertins K., Schmidt G. (eds.) Handbook on Architectures of Information Systems. International Handbook on Information Systems. Springer, Berlin, pp. 347–367

Ferstl O. K., Sinz E. J. (2013) Grundlagen der Wirtschaftsinformatik, 7th ed. Oldenbourg, München

Ferstl O. K., Sinz E. J., Bork D. (2016) Tool Support for the Semantic Object Model. In: Karagiannis D., Mayr H. C., Mylopoulos J. (eds.) Domain-Specific Conceptual Modeling. Springer, Berlin, pp. 291–310

Nüttgens M. (2017) EPK. In: Kurbel K., Becker J., Gronau N., Sinz E., Suhl L. (eds.) Enzyklopädie der Wirtschaftsinformatik – Online-Lexikon 7th ed. Oldenbourg, München, 13.9.2013 http://www.enzyklopaedie-der-wirtschaftsinformatik.de Last Access: 25/09/2014

OMG (2017) Business Process Model and Notation (BPMN). Version 2.0.2. 2014 http://www.omg.org/spec/BPMN/2.0.2/ Last Access: 16/11/2017

Pütz C., Sinz E. J. (2010) Model-driven Derivation of BPMN Workflow Schemata from SOM Business Process Models. In: Enterprise Modelling and Information Systems Architectures 5(2), pp. 57–72

Reisig W. (2010) Petrinetze. Modellierungstechnik, Analysemethoden, Fallstudien. Vieweg + Teubner, Wiesbaden

Scheer A.-W. (1998) ARIS – Modellierungsmethoden, Metamodelle, Anwendungen, 3rd ed. Springer, Berlin

Scheer A.-W., Thomas O. (2005) Geschäftsprozessmodellierung mit der ereignisgesteuerten Prozesskette. In: Das Wirtschaftsstudium 34(8-9), pp. 1069–1078

Vossen G., Becker J. (eds.) (1996) Geschäftsprozeßmodellierung und Workflow-Management. Modelle, Methoden, Werkzeuge. Thomson, Bonn

Weske M. (2012) Business Process Management. Concepts, Languages, Architectures, 2nd ed. Springer, Berlin

# What do Business Process Modelling and Super Mario Bros. have in Common? A Games-perspective on Business Process Modelling

Nicolas Pflanzl[*,a], Gottfried Vossen[a,b]

[a] European Research Center for Information Systems (ERCIS), Münster, Germany
[b] Department of Management Systems, University of Waikato, Hamilton, New Zealand

Abstract. *At first glance, it might not seem as if there was a tangible connection between playing a video game such as Super Mario World, and creating a business process model in a respective software. However, this paper argues that business process modelling itself can in fact be considered a game, and thus current issues of business process modeling such as insufficient model quality and unmotivated process modellers can be attributed to problems of the underlying "game design". As a solution, the activity of building tools for business process modeling may also be addressed using game design techniques, thereby allowing the positive impacts and benefits of games on engagement, motivation, training, and performance to be carried over to this non-game context. Such a games-perspective on business process modelling has already been assumed by a small number of researchers, as will be shown through a discussion of related work. Lastly, this paper calls for additional research situated at the intersection between process modelling and games.*

Keywords. Business Process Modelling • Model Quality • Gamification • Serious Games

## 1 Introduction

Business process models are important artefacts for the design, implementation, enactment, and improvement of business processes in the context of Business Process Management (BPM) (Schönthaler et al. 2012). They are created through business process modelling (BPMod), which is often conceptualized as an activity carried out by a small number of experts eliciting requirements from process end-users through interviews and questionnaires. This understanding is slowly changing, with an increasing number of authors stating that BPMod requires the active involvement of *all* stakeholders to be successful (e. g., (Bandara et al. 2005; Brocke et al. 2014)). However, such an inclusive approach to BPMod introduces new challenges, such as motivating the desired contributors to actually participate, providing tools that enable unexperienced novice modellers to contribute with little modelling skills, and ensuring that the quality of the resulting models is high enough for them to be useful (Pflanzl and Vossen 2014).

To develop solutions for these problems, some authors seek to transfer social software and its underlying principles to the BPM domain, which has led to the emergence of Social BPM (Erol et al. 2010). However, little attention has been devoted to another domain which could also make valuable contributions towards solving the aforementioned challenges: digital games. While such games are primarily designed as entertainment media and have historically been seen as unproductive and disconnected from the "real world"—a view going back to the mid-20th century, cf. Caillois (1961) and Huizinga (1949)—they are increasingly being recognized as tools for training and education that propel players towards ever-increasing levels of performance and can motivate them to continue

* Corresponding author.
E-mail. nicolas.pflanzl@wi.uni-muenster.de

playing until there is nothing left to learn (Connolly et al. 2012; Koster 2005; McGonigal 2011). This has led to the emergence of research areas such as *gamification* (Deterding et al. 2011) and *serious games* (Michael and Chen 2005), which seek to harness the potential of games for purposes other than entertainment.

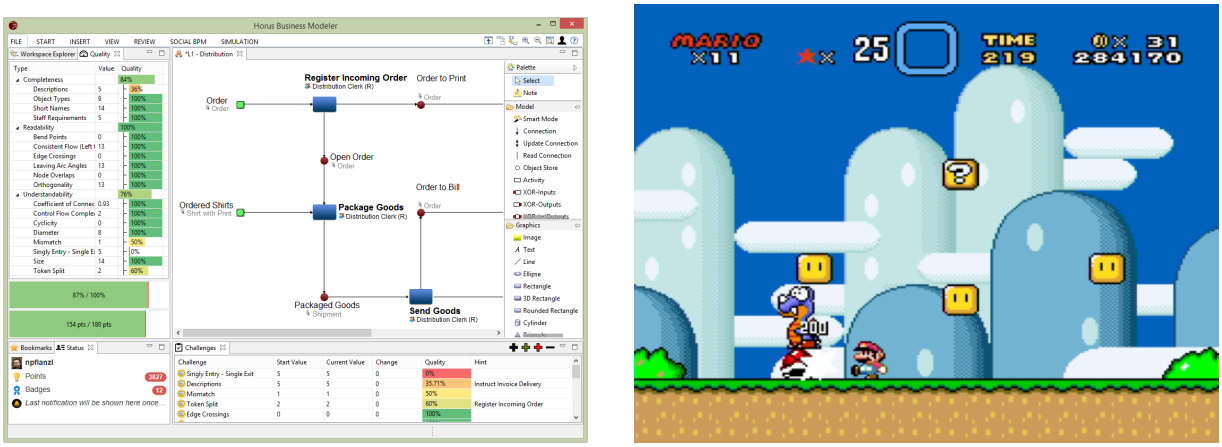
While at first glance it might not seem as if an activity such as modelling business processes (see Figure **??**) could be improved using ideas from playing games like Super Mario World (SMW, see Figure 1b) a direct correspondence between the challenges of the former and the benefits of the latter as outlined in the previous two paragraphs can be observed. This can be seen as an indication that academics should investigate the potential applications of gamification, serious games, and related fields to the business process modelling area. To open an avenue for such research, this paper argues that process modelling itself can already be seen as a sort of game, and that accordingly, many problems the discipline is facing today are the result of inadequate "game design" and implementation, and may be solved (or at least alleviated) using tools and techniques adopted from game design. To that extent, Section 2 will first substantiate this argument by contrasting and comparing BPMod and SMW against a definition of the term "game". Afterwards, Section 3 will demonstrate the relevance of a games-perspective on business process modelling by presenting existing research that has already incorporated this view. The paper ends with a brief summary and a call for future research in Section 4.

## 2 Business Process Modelling as a Game

Despite numerous attempts to define the term "game" (see, e. g., Salen and Zimmerman 2003, Fullerton 2008, Schell 2008), there is still no consensus about its exact meaning and characteristics. However, Juul proposes the following definition based on a synthesis of the works of many other authors: "*A game is a rule-based formal system with a variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort in order to influence the outcome, the player feels attached to the outcome, and the consequences of the activity are optional and negotiable*" (Juul 2011, p. 36). In the following paragraphs, a side-by-side comparison of the game SMW and the activity BPMod will be conducted against the components of this definition to highlight the similarities between both.

**Rules**. The rules are the "core of what a game truly is" (Schell 2008, p. 130) beyond its graphics, technology, and story, and are implemented through its *game mechanics* that specify the actions and behaviours that players may perform (Hunicke et al. 2004). For instance, SMW provides the basic mechanics *walking*, *running*, and *jumping*, and the rules of the game further dictate that players must finish *levels* by reaching their *goals* while overcoming *obstacles* and avoiding or defeating *enemies*. Failing to do so within a given amount of *time* causes players to lose *lives*, which may ultimately result in a *game over*. In the context of BPMod, the rules are imposed by the syntax of the utilised modelling language. For instance, Petri nets are defined as sets of *places*, *transitions*, and *arcs* so that places and transitions are disjoint and arcs may only exist between elements of different sets and thus, e. g., not between to places (Reisig and Rozenberg 1998). The game mechanics are then provided by a particular modelling tool and may consist of operations such as creating, deleting, and modifying model elements.

**Variable and quantifiable outcome**. A game must allow for different outcomes, and players should experience uncertainty about which outcome they are going to attain (Fullerton 2008). Furthermore, outcomes should be unambiguous and beyond discussion, which relies on the calculation of appropriate quantitative measures. For example, when playing a particular level in SMW, players may successfully reach the exit, run out of time, collide with an enemy, or fall down a pit. Further quantification is provided by indicators such as the number of collected coins or the remaining amount of time. In case of BPMod, outcomes comprise the possible models that may be created for a particular modelling task. Since this is an




(a) Horus Business Modeler

(b) Super Mario World

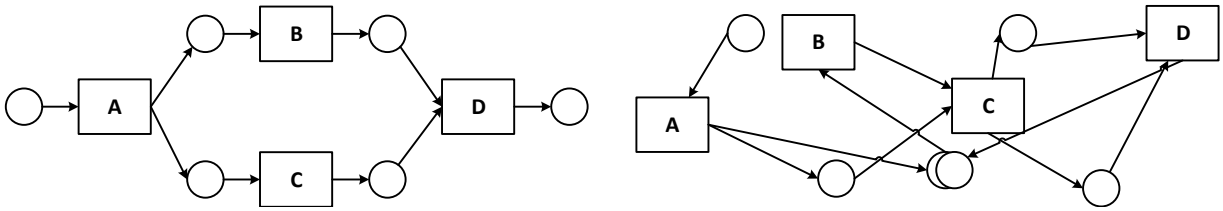
*Figure 1: Modelling a business process (left) vs. playing a video game (right)*

activity carried out by a human modeller based on their experience and perception, any process may be depicted in an infinite number of ways (Becker et al. 2012). These outcomes can then be quantified by means of measurement procedures for various aspects of model quality, such as readability, understandability, and completeness (Overhage et al. 2012).

**Valorisation of outcome**. To each possible outcome, a specific value can be assigned, e. g., a score. Based on this value, some outcomes may be considered "better" than others, and those outcomes with a higher value are typically more challenging to obtain, therefore requiring skill and expertise. For instance, players may finish one particular level of SMW with or without a power-up, and with more (better) or less (worse) collected coins and time remaining. Some levels may even offer a secret exit to be found by particularly skilled players. Similarly, two different representations of the same business process may differ in their value as measured by a set of quality metrics such as those proposed by Mendling (2008). For instance, whereas an experienced process modeller may create a highly readable model as shown in Figure 2a, a different representation of the same process constructed by a modelling novice may contain quality defects such edge crossings and node overlaps (see Figure 2b) that lower its value.

**Player effort**. Games are challenging and allow players to influence the outcome by investing significant effort. As such, they differ from movies and other non-interactive media where the outcome is independent of any player interaction. For example, some levels of SMW may exhibit such a high level of challenge that less experienced players can only complete them after repeated failure. Analogously, BPMod is a highly iterative process in itself, and the initial solution will most commonly not be optimal so that successive refinements must be implemented to achieve a high-value outcome.

**Player attached to outcome**. As a consequence of the invested effort, players feel attached to the result of a game and may experience varying emotional responses based on its outcome. For instance, whereas accomplishing a difficult level in SMW can result in a feeling of pride also referred to as *fiero* (McGonigal 2011), repeated failure may instead lead to frustration and unhappiness. Similarly, it stands to reason that a process modeller may feel, e. g., happy, anxious, or frustrated depending on whether the utilized modelling tools allow them to make the desired statements about a business process. Further emotional attachment




(a) Good configuration (b) Bad configuration

*Figure 2: Quality metrics example: Planar variables*

may be connected to the potential consequences of the modelling endeavour as described next.

**Negotiable consequences**. Based on its outcome, a game may (or may not) have optional consequences in real life that can be negotiated for any given context. This stands in stark contrast to the historic, but obsolete view that playing games is always an inherently unproductive activity that cannot serve any "serious" purpose beyond mere entertainment (Caillois 1961), and that is carried out in a safe environment clearly separated from reality, the so-called *magic circle* (Huizinga 1949; Salen and Zimmerman 2003). An example contradicting this view are the markets that nowadays exist around many games in which players can sell in-game items for money. Playing SMW in turn will most often not entail any real-world consequences, but could also be carried out as a competition or involve a bet with monetary rewards. In case of BPMod, the potential consequences of a process model depend on the purpose for which it was created, such as process improvement, certification, or software development (Becker et al. 2003). However, it may also be the case that a model is only created for documentation purposes, or is rejected due to insufficient quality.

As this discussion illustrates, SMW as well as BPMod may both considered games when evaluated against the definition provided by Juul (2011). Thus, it is reasonable to expect that some of the positive impacts and benefits of digital games may be carried over to the context of process modelling by approaching the creation of respective tools as a game design problem. Such a games-perspective on BPMod has already been adopted by other authors, as the examples presented in the following section will demonstrate.

## 3 Applications

Inspired by the importance of the games industry and the impacts of playing games on motivation, engagement, and skills development (Connolly et al. 2012), researchers are nowadays actively investigating how the power of games can be harnessed to solve real-world problems (McGonigal 2011). This has led to the emergence of many different research areas addressing this topic from varying perspectives, such as gamification (Section 3.1) and serious games (Section 3.2). While the BPMod discipline has been slow to investigate the potential uses of digital games, some initial research can be identified and will be discussed in the remainder of this section.

### 3.1 Gamification

One of the most popular approaches towards exploiting games lies in the integration of elements that are perceived as characteristic for the former into a non-game application domain while maintaining its 'serious" nature. Consequently, while the resulting activity can be more playful, enjoyable, and engaging, it should not be perceived as a game in itself. This approach is called *gamification* and is commonly defined as "the use of game design elements in non-game contexts" (Deterding et al. 2011, p. 10). A typical gamification endeavour consists of identifying goals that are associated with the use of a particular system, quantifying these goals through appropriate metrics, and then

implementing game design patterns such as points, badges, and leaderboards to motivate goal-driven user behaviour (Deterding 2015).

Initial ideas regarding the use of gamification for BPMod can be found in the area of Social BPM, an egalitarian, bottom-up approach towards BPM based on the principles of social software. In this context, EROL ET AL. suggest the use of "honour points" as a means for rewarding users based on their contributions to motivate their voluntary participation (Erol et al. 2010). These points, the authors suggest, could then be exchanged for tangible rewards such as money, acknowledgements, or certifications. Further conceptual work was conducted by RITTGEN, who proposes that BPMod sessions could also be conducted as competitive games in which a specific modelling task is given, participants create competing models, score each other, and the model with the highest score is chosen as the "winner" (Rittgen 2010). The author argues that this may serve as a source of extrinsic motivation, but does not further elaborate on the details of the scoring mechanism.

Expanding upon these theoretical considerations, other authors have described implementations of "gamified" BPMod with varying degrees of sophistication and levels of detail. For instance, AWAD ET AL. describe *ISEAsy*, a software for end-user process modelling that includes experience points and a level system (Awad et al. 2013). However, the authors neither provide information about the activities for which points are awarded, nor the benefits of gaining a level. Furthermore, the superficial discussion and lack of an evaluation prevents making any conclusions about the effectiveness of the implementation. A more elaborate approach and corresponding software prototype are outlined by HOPPENBROUWERS AND SCHOTTEN based on an interpretation of BPMod as a game with the following mechanics: modelling goals, immediate audio-visual feedback, and a score that reduces over time, thereby causing time pressure (Hoppenbrouwers and Schotten 2009). The proposed score system rewards modellers with 100 points for defining process activities, 100 points for creating control flow arcs, and 10 points for the definition of input and output objects. While the authors include game design elements as a foundation rather than secondary components, it should be noted that their scoring scheme is limited by rewarding quantity of work instead of quality.

The most extensive implementation of a gamified process modelling tool to date was presented by PFLANZL ET AL. (Pflanzl 2016; Pflanzl et al. 2017). Based on a review of literature on the quality of business process models, the prototype called *Horus Gamification* (see Figure 1a) implements a set of quality metrics addressing the readability, understandability, and completeness of process models. These metrics are the foundation for a scoring mechanism that rewards users for the quality of their models. Furthermore, users receive real-time quality feedback, have the possibility to unlock badges for notable behavior, and can compete with others on a points-based leaderboard. The implemented prototype was used in a field study with first-semester Information Systems students who were randomly assigned to either the experimental group (with gamification) or the control group (without gamification). The data obtained from the study demonstrates that gamified BPMod can lead to a statistically significant improvement of model quality for all three categories of metrics (Pflanzl 2017).

### 3.2 Serious Games

As indicated by the name, serious games are (digital or non-digital) games that are designed for a primary purpose other than entertainment (Michael and Chen 2005). They differ from gamified applications in that they are designed and experienced as full-fledged games rather than non-game systems that just contain elements of the games.

Very little research can be identified at the intersection between BPMod and serious games. One example is the work conducted by BROWN ET AL. who examine the use of virtual worlds as an environment for distributed, collaborative modelling sessions involving both experts and novice users (Brown 2010; Brown et al. 2011). Through this, BPMod is effectively rendered a

game that is played in a 3D environment with opportunities for additional immersion through, e. g., virtual reality headsets. Another example is Innov8 2.0, a game that was developed by IBM as a tool for educating players about the organizational benefits and importance of BPMod (Blohm and Leimeister 2013). In this game, players act as consultants and are tasked with increasing the effectiveness and efficiency of certain business processes through process re-engineering (Sumarie and Joubert 2009).

## 4 Conclusion and Outlook

Overall, the presented applications show that game-based ideas are slowly being incorporated into BPMod research, although in a more cautious fashion than in other disciplines. Consequently, many of the presented concepts remain superficial (e. g., by focusing on work quantity instead of quality) and include game elements as shallow add-ons rather than as an integral part of the underlying system's fabric. Furthermore, the lack of empirical data and experience reports means that any claims about the potential impacts of gamification and serious games for BPMod remains speculation and conjecture.

Beyond process modeling, some researchers have also proposed solutions inspired by game design for other aspects of the BPM life cycle, such as using gamification to educate novices about the use of a process modeling language (De Smedt et al. 2016), building a gamified and competitive system for specifying rewards and incentives for business process execution in crowdsourced settings (Scekic et al. 2012), and exploiting serious games as a tool for training end-users in process enactment (Pflanzl et al. 2016).

In conclusion, the current state of the art offers numerous possibilities for research at the intersection between process modelling and games, such as 1) examining more sophistiacted ways of gamifying existing BPMod tools that go beyond the mere integration of points, badges, and leaderboards; 2) designing and implementing serious games that teach players process modelling skills through their gameplay; 3) generating playable serious games out of reference models to provide process end-users with gameful process training; 4) studying potential uses of commercial off-the-shelf games (esp. strategy and city-building games as well as related genres) as BPM teaching tools; 5) and lastly finding new application scenarios in other phases of the BPM life cycle.

## References

Awad A., Decker G., Lohmann N. (2013) ISEAsy: A Social Business Process Management Platform. In: Proceedings of the 6th Workshop on Business Process Management and Social Software (BPMS 2013). Beijing, China, pp. 125–137

Bandara W., Gable G. G., Rosemann M. (2005) Factors and Measures of Business Process Modelling: Model Building through a Multiple Case Study. In: European Journal of Information Systems 14(4), pp. 347–360

Becker J., Kugeler M., Rosemann M. (2003) Process Management: A Guide for the Design of Business Processes. Springer, Berlin, Heidelberg, Germany

Becker J., Probandt W., Vering O. (2012) Grundsätze Ordnungsmäßiger Modellierung: Konzeption und Praxisbeispiel für ein Effizientes Prozessmanagement, 1st. Springer, Berlin, Heidelberg, Germany

Blohm I., Leimeister J. M. (2013) Gamification: Design of IT-Based Enhancing Services for Motivational Support and Behavioral Change. In: Business & Information Systems Engineering 5(4), pp. 275–278

vom Brocke J., Schmiedel T., Recker J. C., Trkman P., Mertens W., Viaene S. (2014) Ten Principles of Good Business Process Management. In: Business Process Management Journal 20(4), pp. 530–548

Brown R. A. (2010) Conceptual Modelling in 3D Virtual Worlds for Process Communication. In: Proceedings of the 7th Asia-Pacific Conference on Conceptual Modeling (APCCM 2010). Brisbane, Australia, pp. 25–32

Brown R. A., Recker J. C., West S. (2011) Using Virtual Worlds for Collaborative Business Process Modeling. In: Business Process Management Journal 17(3), pp. 546–564

Caillois R. (1961) Man, Play and Games, 1st. University of Illinois Press, Champaign, IL, United States

Connolly T. M., Boyle E. A., MacArthur E., Hainey T., Boyle J. M. (2012) A Systematic Literature Review of Empirical Evidence on Computer Games and Serious Games. In: Computers & Education 59(2), pp. 661–686

De Smedt J., De Weerdt J., Serral E., Vanthienen J. (2016) Gamification of Declarative Process Models for Learning and Model Verification. In: Proceedings of the 3rd International Workshop on Decision Mining & Modeling for Business Processes (DeMiMoP 2015). Innsbruck, Austria, pp. 432–443

Deterding S. (2015) The Lens of Intrinsic Skill Atoms: A Method for Gameful Design. In: Human-Computer Interaction 30(3-4), pp. 294–335

Deterding S., Dixon D., Khaled R., Nacke L. E. (2011) From Game Design Elements to Gamefulness: Defining "Gamification". In: Proceedings of the 15th International Academic MindTrek Conference (MindTrek 2011). Tampere, Finland, pp. 9–15

Erol S., Granitzer M., Happ S., Jantunen S., Jennings B., Johannesson P., Koschmider A., Nurcan S., Rossi D., Schmidt R. (2010) Combining BPM and Social Software: Contradiction or Chance? In: Journal of Software Maintenance and Evolution: Research and Practice 22(6-7), pp. 449–476

Fullerton T. (2008) Game Design Workshop: A Playcentric Approach to Creating Innovative Games, 1st. Morgan Kaufmann Publishers, Burlington, MA, United States

Hoppenbrouwers S. J. B. A., Schotten B. (2009) A Game Prototype for Basic Process Model Elicitation. In: Proceedings of the 2nd IFIP Working Conference on The Practice of Enterprise Modeling (PoEM 2009). Stockholm, Sweden, pp. 222–236

Huizinga J. (1949) Homo Ludens: A Study of the Play-Element in Culture, 2nd. Routledge & Kegan Paul, London, United Kingdom

Hunicke R., LeBlanc M., Zubek R. (2004) MDA: A Formal Approach to Game Design and Game Research. In: Proceedings of the 2004 AAAI Workshop on Challenges in Game Artificial Intelligence. San Jose, CA, United States

Juul J. (2011) Half-Real: Video Games between Real Rules and Fictional Worlds. MIT Press, Cambridge, MA, United States

Koster R. (2005) A Theory of Fun for Game Design, 1st. Paraglyph Press, Scottsdale, AZ, United States

McGonigal J. (2011) Reality is Broken: Why Games Make Us Better and How They Can Change the World, 1st. The Penguin Press, New York City, NY, United States

Mendling J. (2008) Metrics for Process Models: Empirical Foundations of Verification, Error Prediction, and Guidelines for Correctness, 1st. Springer, Berlin, Heidelberg, Germany

Michael D., Chen S. (2005) Serious Games: Games that Educate, Train, and Inform, 1st. Thomson Course Technology, Boston, MA, United States

Overhage S., Birkmeier D. Q., Schlauderer S. (2012) Quality Marks, Metrics, and Measurement Procedures for Business Process Models: The 3QM-Framework. In: Business & Information Systems Engineering 4(5), pp. 229–246

Pflanzl N. (2016) Gameful Business Process Modeling. In: Proceedings of the 7th International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA 2016). Vienna, Austria, pp. 17–20

Pflanzl N. (2017) Gamification for Business Process Modeling. Doctoral Thesis, University of Münster

Pflanzl N., Classe T., Araujo R., Vossen G. (2016) Designing Serious Games for Citizen Engagement in Public Service Processes. In: Proceedings of the 9th Workshop on Business Process Management and Social Software (BPMS2 2016). Rio de Janeiro, Brazil, pp. 1–12

Pflanzl N., Vetter A., Hussong K., Oberweis A., Schönthaler F., Vossen G. (2017) Horus Gamification: Ein Prototyp zum Einsatz von Gamification im Rahmen des Social BPM. In: Proceedings of the 47th Conference of the Gesellschaft für Informatik (INFORMATIK 2017). Chemnitz, Germany, pp. 433–438

Pflanzl N., Vossen G. (2014) Challenges of Social Business Process Management. In: Proceedings of the 47th Hawaii International Conference on System Sciences (HICSS 2014). Waikoloa, HI, United States, pp. 3868–3877

Reisig W., Rozenberg G. (1998) Lectures on Petri Nets I: Basic Models, 1st. Springer, Berlin, Heidelberg, Germany

Rittgen P. (2010) Success Factors of e-Collaboration in Business Process Modeling. In: Proceedings of the 22nd International Conference on Advanced Information Systems Engineering (CAiSE 2010). Hammamet, Tunisia, pp. 24–37

Salen K., Zimmerman E. (2003) Rules of Play: Game Design Fundamentals, 1st. The MIT Press, Cambridge, MA, United States

Scekic O., Truong H.-L., Dustdar S. (2012) Modeling Rewards and Incentive Mechanisms for Social BPM. In: Proceedings of the 10th International Conference on Business Process Management (BPM 2012). Tallinn, Estonia, pp. 150–155

Schell J. (2008) The Art of Game Design: A Book of Lenses, 1st. Morgan Kaufmann Publishers, Burlington, MA, United States

Schönthaler F., Vossen G., Oberweis A., Karle T. (2012) Business Processes for Business Communities: Modeling Languages, Methods, Tools, 1st. Springer, Berlin, Heidelberg, Germany

Sumarie R., Joubert P. (2009) Evaluating Serious Games in Higher Education: A Theory-based Evaluation of IBMs Innov8. In: Proceedings of the 3rd European Conference on Games Based Learning (ECGBL 2009). Graz, Austria, pp. 332–338

# A Process Warehouse Model Capturing Process Variants

Lisana Berberi[a], Johann Eder[*,a], Christian Koncilia[a]

[a] Department of Informatics-Systems, Alpen-Adria Universität Klagenfurt, Austria

Abstract. *Process Warehouses are a well-established means for analysing the execution of business processes and the computation of key performance indicators. We propose a new model for process warehouses which is better suited to cope with business processes which have many variants. We present a meta-model of processes with a notion of generic activities which then is used to automatically generate a generalisation hierarchy for process variants along which OLAP operations can be performed.*

Keywords. Business Process Model • Process Warehouse • Process Variant

## 1 Introduction

Process Warehouses (Benker 2016; Eder et al. 2002; Pau et al. 2007) are an appropriate means for analysing the performance of business process execution using well established data warehouse technology and on-line analytical processing (OLAP) tools. In particular, they allow the definition, computation and monitoring of key performance indicators along several dimensions. Typical dimensions in process warehouses are process, time, actor, geographic location. While most of the dimensions are organized in hierarchies supporting roll-up and drill-down operations, the process dimension usually is relatively flat, often comprising just two levels: activity and process, sometimes augmented with a part-of hierarchy but typically without a generalization hierarchy.

Some aspects of processes are still poorly supported such as processes with variants (Döhring et al. 2014) or interorganisational processes (Groiss and Eder 1997). Frequently processes exist in several different variants or versions within the same enterprise and even more so between enterprises. These variants are due to different regulations in different countries, variations due to different requirements for different branches of an enterprise (Kop et al. 2005; Mayr et al. 2007) or different decision histories and responsibilities. Variants also arise due to process evolution and the arising differences add additional complexity to modelling temporal data warehouses (Eder et al. 2001). Even if all these variants were expressed in a single process definition (with the excessive use of xor-splits), the resulting processes are large, difficult to understand and to communicate and overloaded, and new process definitions still comprise of all the past processes definitions they should replace. Analysing sets of process variants is cumbersome with current process warehouse technology.

In this paper we propose the core of a process warehouse model with a generalization hierarchy of processes which captures process variants. This generalization hierarchy can be generated from a meta-model of business process models which introduces the notion of generic activities which generalize a set of activities (e. g., pay by credit card, by check, or by third-party (PayPal) could all be generalized to an activity payment).

Based on given hierarchies of activities we define generalizations of processes for the "process" dimension of a process warehouse. This hierarchy can the be used to roll-up or drill down when analysing the logs of the executions of the various process variants and it makes it much easier to compare key-performance indicators between different variants at different levels of genericity.

* Corresponding author.
E-mail. johann.eder@aau.at

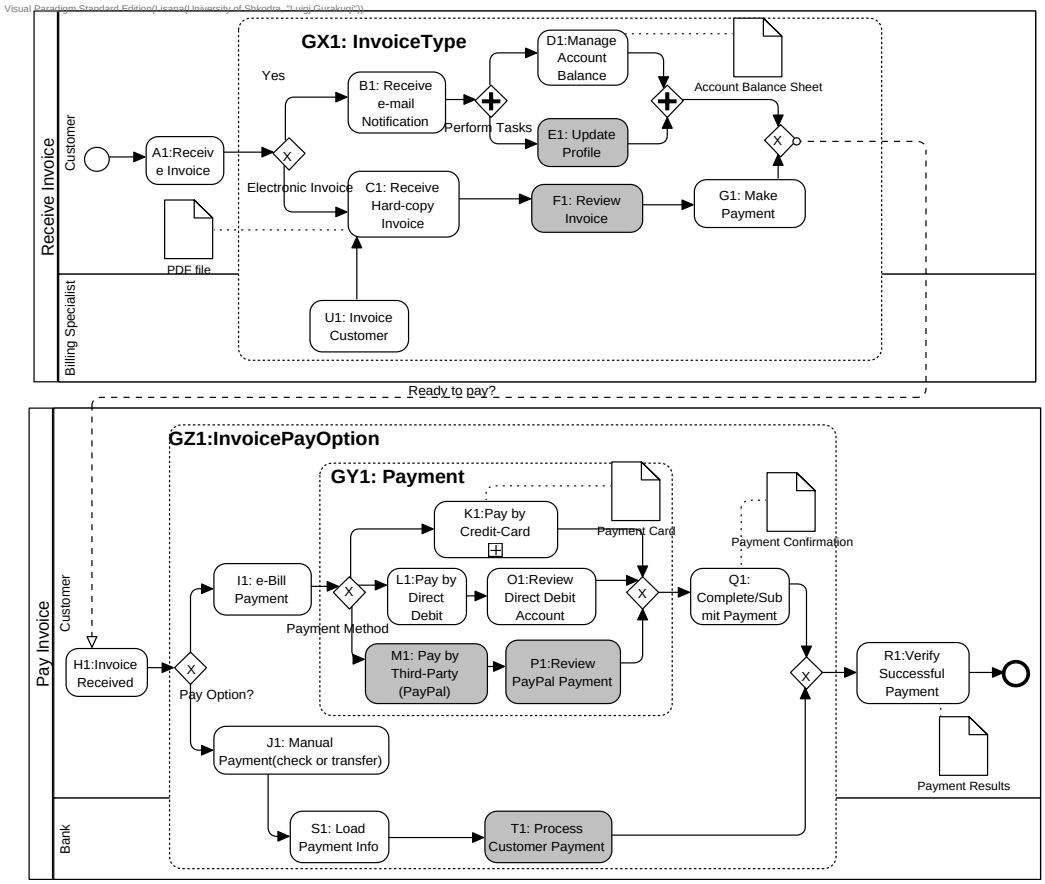



*Figure 1: Processing Customer Invoice Payment in Texas/USA.*

## 2 Motivating Example

Let's assume we have a core business process, e. g., Processing Customer Invoice Payments of a financial administration agency that is modelled as an interaction between two processes named *ReceiveInvoice* and *PayInvoice*.

*ReceiveInvoice* process consists of a set of activities that checks if a customer has requested electronic or hard-copy invoice for goods or services, while *PayInvoice* process consists of a set of other activities that submit or complete with the payment after a customer invoice is received.

We illustrate our approach of generic processes and generic activities with 2 variants of this business process in two different states Texas and New York as shown in Figures 1 and 2. The differences between variants are highlighted with light gray.

Both variants start with Activity *A1: Receive Invoice* followed by a decision point (depicted with an X diamond) where one of the outgoing activities i. e., *B1: Receive e-mail Notification* or *C1: Receive Hard-copy Invoice* is to be executed depending on the type of the invoice.

After receiving e-mail notification two parallel activities should be executed: *D1: Manage Account Balance* or *E1: Update Profile* in variant 1 whereas in variant 2 *D2: Manage Account Balance* and *E2: Review Payment History*.

In variant 1 if the received invoice is a hard-copy invoice which is issued by a billing specialist when activity *U1: Invoice Customer* is enacted. Afterwards *F1: Review Invoice* followed by *G1: Make Payment* is executed.

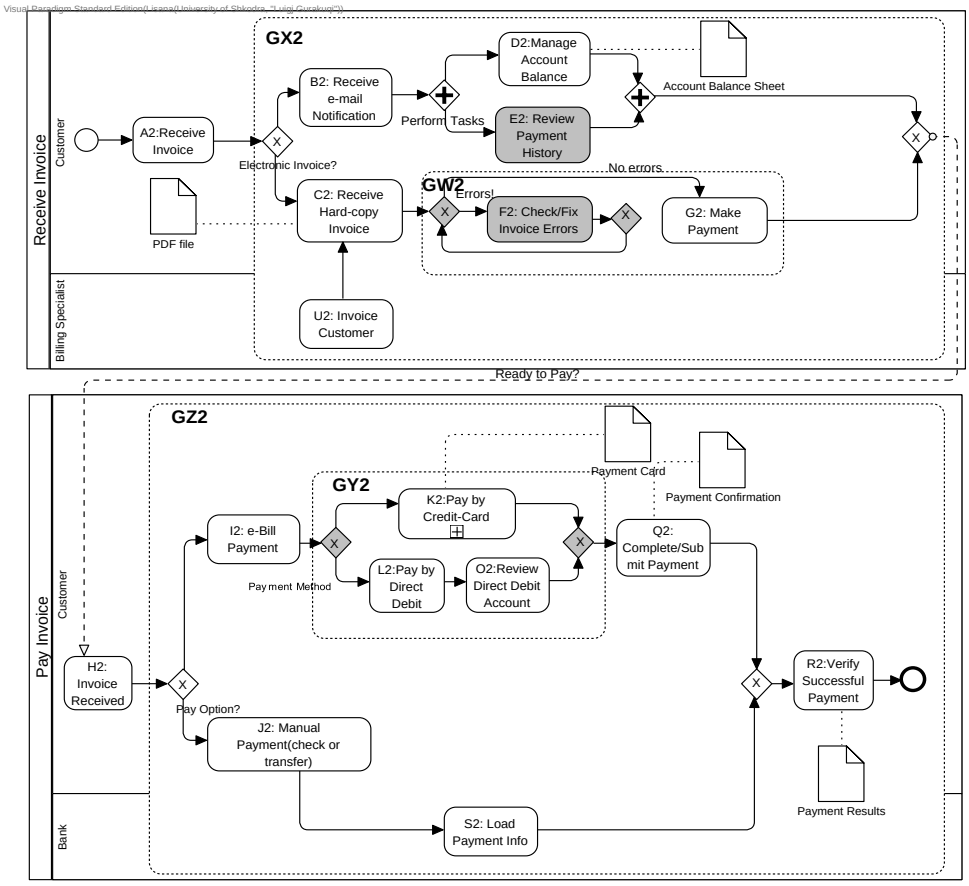



*Figure 2: Processing Customer Invoice Payment in New York/USA.*

Whereas, in variant 2 if the received invoice is a hard-copy invoice then a loop decision point that checks for the invoice errors is executed. If no errors are found then the flow is shifted to the second process *PayInvoice*, otherwise *F2: Check/Fix Invoice Errors* is executed several times until the invoice errors are fixed.

The invoice may be paid manually or electronically, depending on this option two activities might be executed: *I1: e-Bill Payment* or *J1: Manual Payment (check or transfer)* in both variants. If electronic payment is chosen, then in variant 1 three possibilities are offered *(K1: Pay by Credit Card* or *L1: Pay by Direct Debit* followed by *O1: Review Direct Debit Account* or *M1: Pay by Third-Party (PayPal)* followed by *P1: Review PayPal Payment)* and only two in variant 2 (pay by credit card or by direct debit).

The sub process *K1: Pay by Credit Card* checks if it's enough credit to pay the invoice order. If yes, then activity Charge credit is executed otherwise Notify client is executed. But, if manual payment is chosen then in variant 1 activity S1: Load Payment Info is executed from a bank representative, followed by *T1: Process Customer Payment*, whereas only activity *S2: Load Payment Info* is executed in variant 2. After selecting the payment method activity *Q1: Complete/Submit Payment* is executed, a payment confirmation document is generated.

Finally, both variants end with executing *R1: Verify Successful Payment* activity.

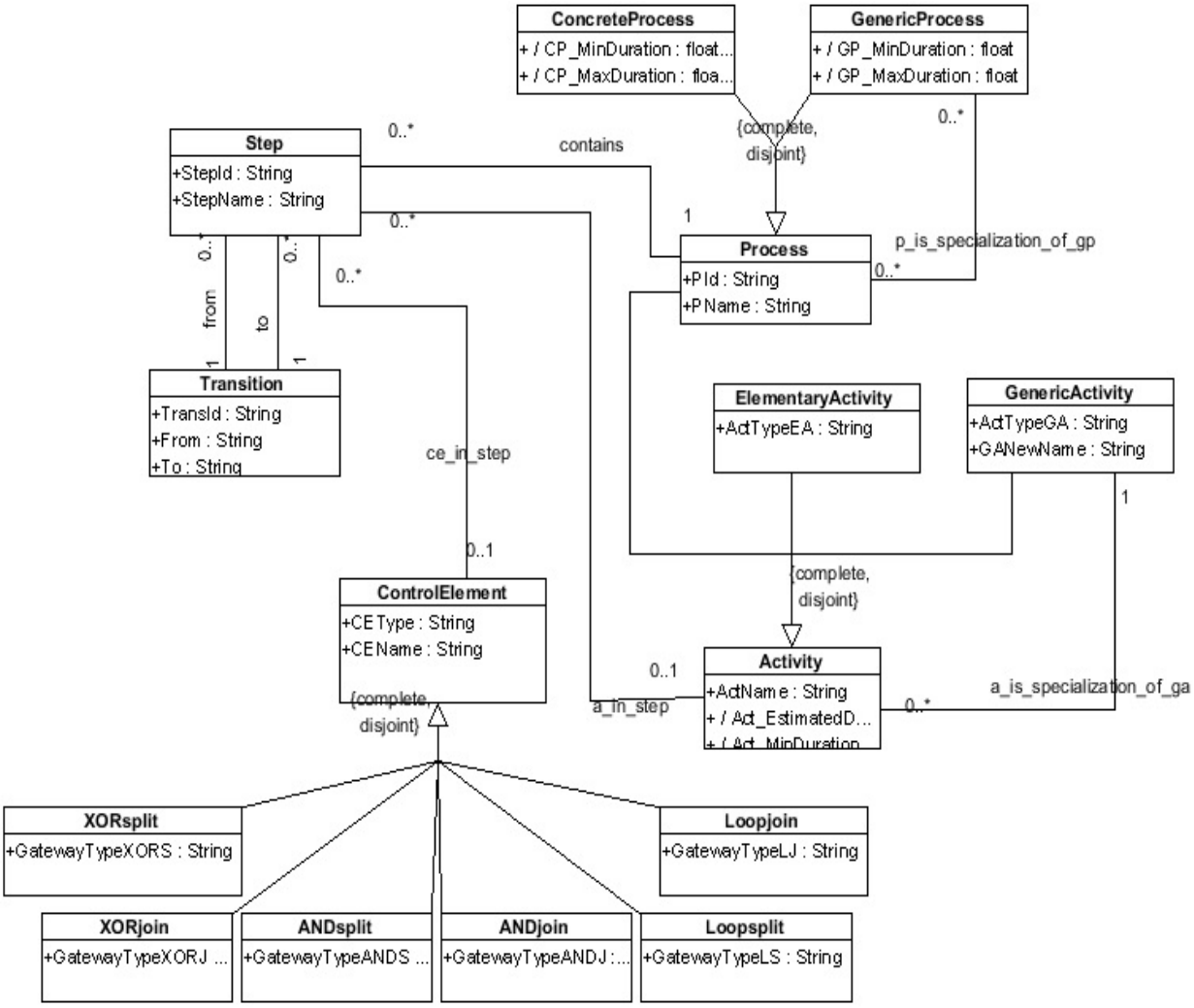



*Figure 3: Process meta-model capturing modelling elements.*

## 3 Process Meta-Model with Generic Activities

A *process* is a collection of activities, participants, and dependencies between activities. Activities correspond to individual steps in a business process, participants (software systems or users) are responsible for the enactment of activities, and dependencies determine the execution sequence of activities and the data flow between them (Eder et al. 2002). The process meta-model shown in Figure 3 captures process model elements and their variants from a design-time perspective. A *step* is a concrete invocation of activity in a process and it can be either *Activity* (e. g., activity A) or *ControlElement* (e. g., XORSplit). Each step has different predecessors (from-relationship) and successors (to-relationship), which is expressed by the association class Transition.

An *activity* (the smallest unit of work scheduled by a workflow engine during process enactment) can be of the following subtypes: *elementary activity*, *generic activity* or *process (sub-process)*, where each of these elements are represented through respective classes in the meta-model in a generalization hierarchy (cf. Fig.3). An elementary activity is an atomic/uncompounded activity,

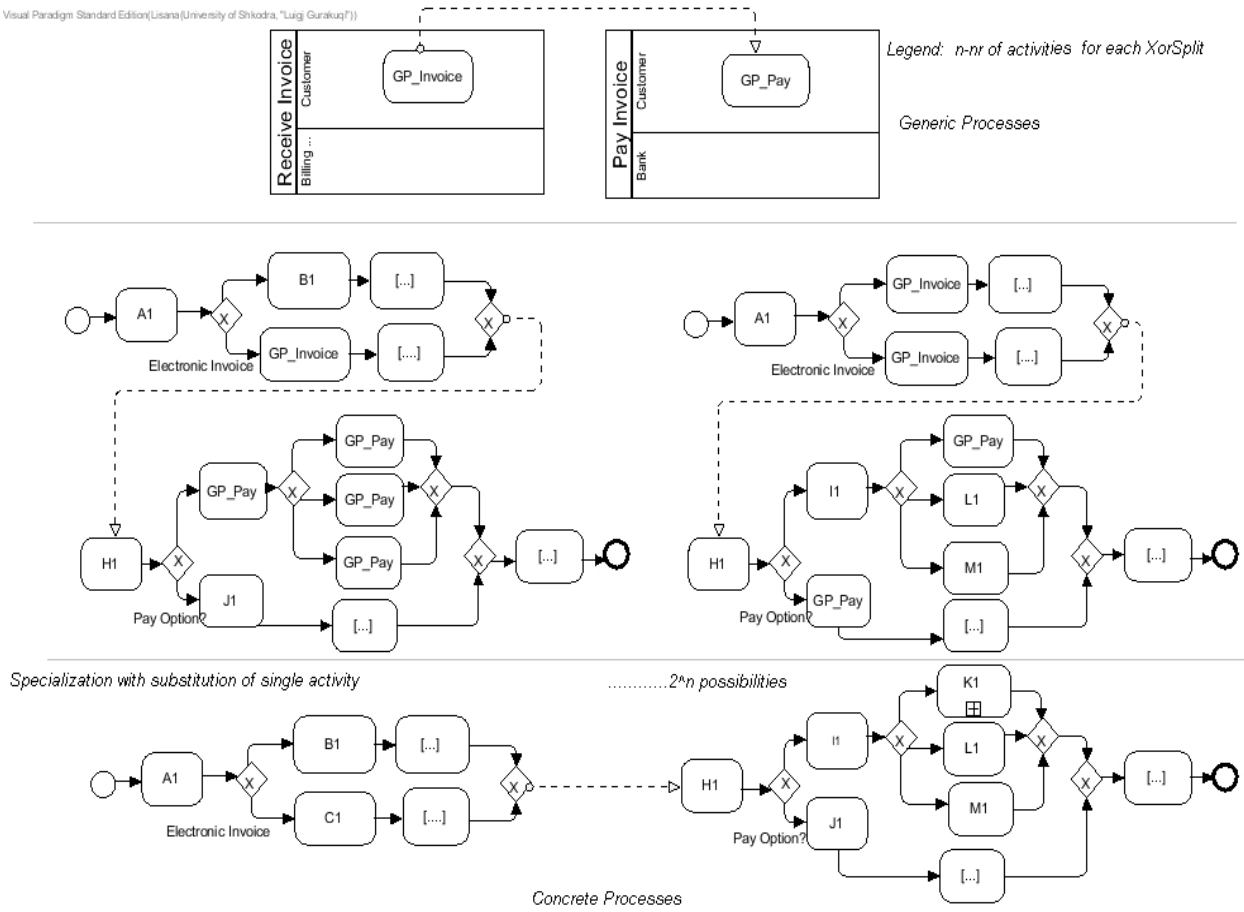



*Figure 4: An excerpt of specialization from Generic Processes to Concrete Processes.*

e. g., activity *A1: Receive Invoice*. A *sub-process* (complex activity) is composed of other activities, e. g., *K1: Pay by Credit-Card* composed of activities *KA1: Check Credit*, *KB1: Charge Credit* and *KC1: Notify Client*. A *process* in a process definition consists of many steps which are logically related in order to achieve a common goal, represented with class *ConcreteProcess (CP)* in our model. Here, we introduce two notions: generic activities and generic processes.

A *generic activity (GA)* is defined as a step in a process that might be realized by different activities, e. g., GX1, GX2 as depicted with a dotted line rectangular in Figs. 1and 2. In the metamodel a generic activity and its specializations are related by 'a_is_specialization_of_ga' in Fig. 3. For example, the subprocess in Fig. 1 Electronic Invoice and activities B, C are specializations of the generic activity GX1.

A *generic process (GP)* is defined as a process that contains at least one generic activity, e. g., GP_Pay consists of generic activities GY1 and GZ1 as shown in Fig. 1. CP and GP are modelled as disjoint subtypes of Process super type class, thus a CP cannot contain any GA.

A *substitution* replaces a generic activity g in a process model with one of the activities a, which are specializations of g. A process P is a *specialization* of a generic process G, if P can be derived from G by substituting one of the generic activities g of G with one of g's specializations. e. g., the process *PayInvoice* is a specialization of the generic process GP_Pay. Fig. 4 shows

specializations by substitution of a single activity from GP to CP.

For the purposes of capturing process variants, the identification of multiple appearances of the same activity is very important, as it allows to aggregate measures of the same activity in different positions of different workflow variants.

## 4 The Generic Process Warehouse Model for Process Variants

Process warehouses typically feature the dimensions process, organization (department hierarchy), actor, geolocation, time. The process dimension usually only represents the part-of decomposition of processes. With the introduction of generic processes as discussed in the preceding section we are able to also provide consolidation hierarchies for process variants. The PWH schema is derived based on the workflow meta model. A Process dimension will be derived with respective attributes (cf. section (a) of Fig. 5) that stores information about which are the steps, activities, processes that are planned for a process enactment to achieve a specific goal. Some other typical dimensions from resource or organizational perspectives might be Participant (alternatively named agent) and Geolocation. We decided to separate these two dimensions to express the fact that different users can belong to different departments in different processes.

Participant stores information about users (human or system) with specific role, e. g., billing specialist, that are responsible for the execution of activities. Geolocation stores information about the geographic locations of a branch structured in different organization units (departments) where a user belongs. Time is another important dimension for our process warehouse that stores information about time needed to execute an activity etc. Hierarchy consolidation paths from the highest level to the lowest one are defined for every dimension table, to show the relationship between their relative attributes. At the lowest level of process dimension is step within the workflow schema. To consolidate the steps to the relevant activities and to consolidate the activities to the corresponding processes of different variants, two types of hierarchies are defined (cf. Fig. 5).

In process dimension the hierarchy coloured with light gray colour express the fact that a step is rolled-up to an activity, then an activity to a generic activity, whereas the hierarchy coloured with light blue colour express the fact that a step is rolled-up to a process, then a process to a generic activity or to a generic process. An example of instances (process attributes members) is shown in section (b), i. e., some possible execution paths (an excerpt of them) e. g., 'A1⟶B1⟶[D1E1]⟶H1⟶I1⟶K1⟶Q1⟶R1' is rolled-up to higher levels 'A1⟶B1⟶[D1E1]⟶H1⟶GZ1⟶R1' (cf. from light gray coloured hierarchy) and 'A1⟶GX1⟶GP_Pay' (cf. from from light blue coloured hierarchy) etc., and then every of them is rolled-up to the highest level that contains only generic processes, i. e., 'GP_Invoice⟶GP_Pay'.

Time has two hierarchies named month and week hierarchy with specific consolidation paths e. g., month hierarchy where a day is rolled-up to a month, a month to a quarter and a quarter to a year, as shown in section b of Figure 5.

The *Participant* dimension with the lowest level of the participant him/herself is consolidated to a combination of users and roles, expressing the fact that a user can have a specific role in an organization unit. The *Geolocation* dimension with the lowest level organization unit (OU) consolidated to a super organization unit (superOU), e. g., Management Department can have a higher hierarchy Financial Department. From superOU we consolidate to a branch, a branch to a city and a city to a country to express the fact that different departments can be part of different branches, and different branches can be located to different cities and to different countries, e. g., the customer payment process in USA and Canada.

So, typical OLAP roll-up and drill down operations can be performed, i. e., from a specific activity or a process we can roll-up to a generic activity and the other way round, from generic activity we can drill-down to activity or process.

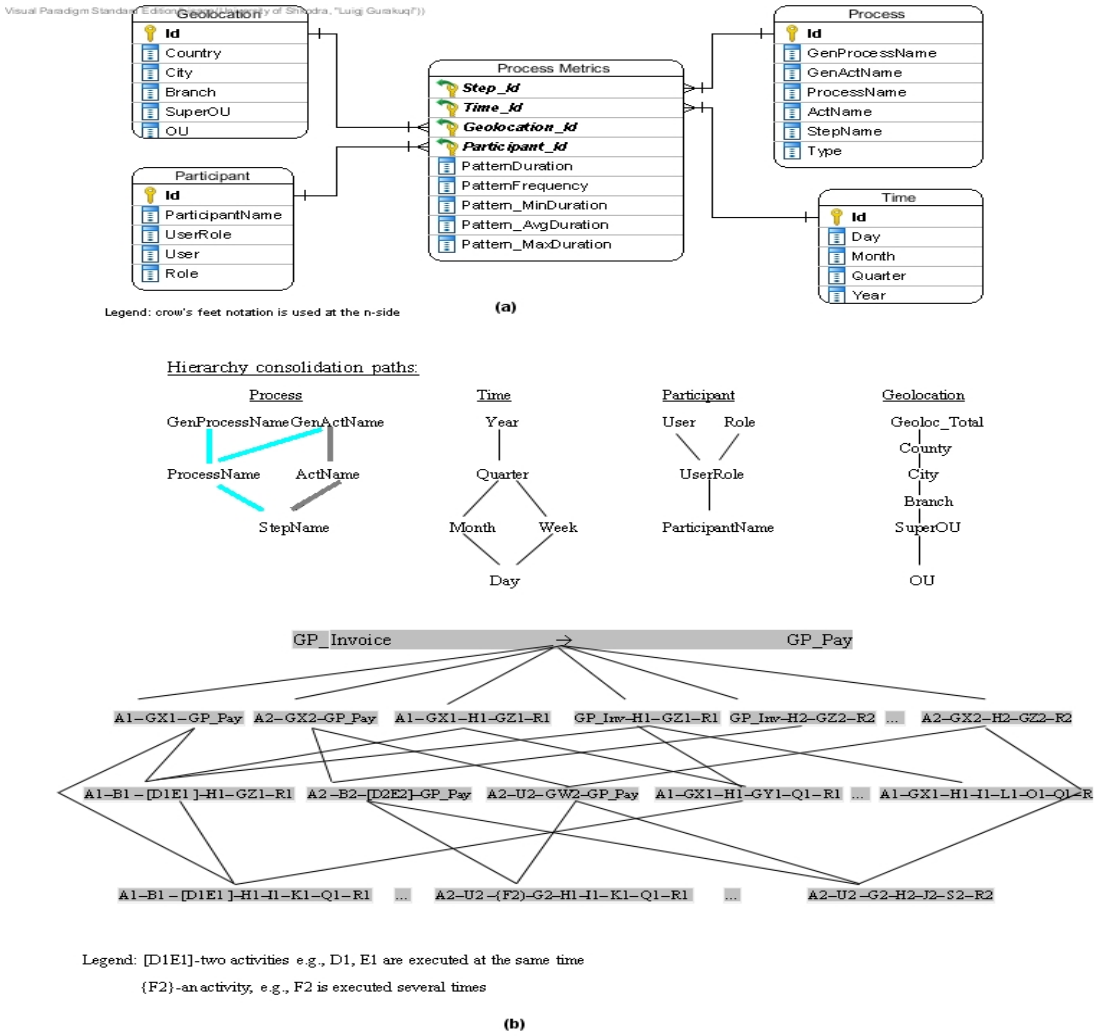



*Figure 5: The process warehouse: (a) star schema and (b) example of instances.*

From a process we can roll-up to a generic process and drill down the other way round.

Now we can express structural and complexity metrics in process models, called process metrics, e. g., key performance indicator (KPI) for measuring the duration (in time units) or cost of each process patterns (path) along different variants, i. e., Duration of process path starting from activity H1 if customer selected to pay by credit-card and ended to activity R1 (H1⟶I1⟶K1⟶Q1⟶R1). Other KPIs might be defined to calculate the most frequent pattern/behaviour after a split condition type is selected in process variants. The generic process structure consolidates different variants in a multidimensional way, e. g., to derive the average, minimum and maximum duration of payments across all different variants of the payment process.

## 5 Related Work

Enhancing analysis of business processes by employing data warehousing and data mining technologies has attracted research over the last 15 years, however, the problem of how to deal with process variants is still not covered satisfactorily.

Some studies (Eder et al. 2002; Koncilia et al. 2015; List et al. 2002; Pau et al. 2007) have been proposed on designing data warehouses for workflow audit trails called 'workflow logs' to exploit the valuable information they contain. (Eder et al. 2002; Pau et al. 2007) construct the respective logical data warehouse models using ADAPT (Application Design for Analytical Processing Technologies) notation.

(Benker 2016) recently proposes to derive data warehouse structures from the meta model of BPMN in order to be portable between application domains and to be stable in case of changing workflows. The data warehouse schema, however, does not support analysis of process variants.

Approaches like (Niedrite et al. 2007; Shahzad and Zdravkovic 2012) presented a goal-driven (i. e., a goal is a state of a process in terms of the quality of the service property that is intended to be achieved) requirement analysis together with the method to obtain the conceptual model of a data warehouse.

To apply state-of-the-art analysis in different workflow application domains, especially in Surgical Workflows, multidimensional modelling seems a promising solution as it allows viewing data from different perspectives and at different granularities. (Mansmann et al. 2007a,b) designed a surgical process recording scheme as UML class diagram. For the convergence of business process model and multidimensional model they found common abstraction between them.

A generic solution for warehousing business processes validated over HP and its customer processes is proposed in (Casati et al. 2007). They abstracted process models and then mapped the process progression (i. e., associating the start and the completion of each step) to events occurring in the source systems.

(Koncilia et al. 2015) focused on analysing complex workflow logs, i. e., different splits/joins and loops, by means of OLAP tools. They defined different sequence of events captured from the trace Log after importing into the DW.

Recently, process mining approaches propose techniques for multiple inter-related processes that may change over time (Aalst 2013) in form of process cube constructed by 3 dimensions, i. e., class type, event class, and time window. In (Liu et al. 2011) a novel E-Cube model is presented which combines complex event processing (CEP) and OLAP techniques for efficient multi-dimensional event pattern analysis at different abstraction levels. They built an event pattern hierarchy that integrates complex event patterns specified using sequence, negation and concept abstractions.

A multidimensional event log (MEL) analysis is proposed by (Vogelgesang and Appelrath 2015) which maps the structure of event logs to a data cube and organizes instances i. e., cases and events on different levels.

An evaluation of various process warehousing approaches through a comprehensive survey is discussed by (Shahzad and Johannesson 2009).

## 6 Conclusions

We argued that process variants can be efficiently and effectively analysed with a process warehouse using specialization/generalization hierarchies between processes and activities. The introduction of a genericity relationship between activities and generic activities allows to generate such generalization hierarchies for processes to structure the "process" dimension of process warehouses, which then can be used to analyse process metrics with the usual OLAP operations such as to roll-up and drill down the dimension hierarchy. In particular it allows to analyse variants of the same process as individuals together, or partitioned in similarity groups.

## References

van der Aalst W. M. P. (2013) Process Cubes: Slicing, Dicing, Rolling Up and Drilling Down Event Data for Process Mining. In: Asia Pacific Business Process Management. Springer Verlag, pp. 1–22

Benker T. (2016) A Generic Process Data Warehouse Schema for BPMN Workflows In: Business Information Systems, BIS, Proceedings Springer International Publishing, pp. 222–234

Casati F., Castellanos M., Dayal U., Salazar N. (2007) A Generic Solution for Warehousing Business Process Data. In: Proceedings of the 33rd International Conference on Very Large Data Bases. VLDB Endowment, pp. 1128–1137

Döhring M., Reijers H. A., Smirnov S. (2014) Configuration vs. adaptation for business process variant maintenance: an empirical study. In: Information Systems 39, pp. 108–133

Eder J., Koncilia C., Morzy T. (2001) A Model for a Temporal Data Warehouse. In: Open enterprise solutions: systems, experiences and organizations Workshop (OES-SEO 2001), pp. 48–54

Eder J., Olivotto G. E., Gruber W. (2002) A Data Warehouse for Workflow Logs. In: Proc. Int. Conf. on Engineering and Deployment of Cooperative Information Systems. Springer, pp. 1–15

Groiss H., Eder J. (1997) Workflow systems for inter-organizational business processes. In: SIGGroup Bulletin 18, pp. 23–26

Koncilia C., Pichler H., Wrembel R. (2015) A Generic Data Warehouse Architecture for Analyzing Workflow Logs. In: Advances in Databases and Information Systems. Springer, pp. 106–119

Kop C., Vöhringer J., Hölbling M., Horn T., Mayr H. C., Irrasch C. (2005) Tool Supported Extraction of Behavior Models.. In: ISTA Vol. 63, pp. 114–123

List B., Schiefer J., Tjoa A. M., Quirchmayr G. (2002) Multidimensional business process analysis with the process warehouse. In: Knowledge Discovery for Business Information Systems. Springer, pp. 211–227

Liu M., Rundensteiner E., Greenfield K., Gupta C., Wang S., Ari I., Mehta A. (2011) E-Cube: Multi-dimensional Event Sequence Analysis Using Hierarchical Pattern Query Sharing. In: Proc. of the 2011 ACM SIGMOD International Conference on Management of Data. ACM, pp. 889–900

Mansmann S., Neumuth T., Scholl M. H. (2007a) Multidimensional data modeling for business process analysis. In: Conceptual Modeling-ER 2007. Springer, pp. 23–38

Mansmann S., Neumuth T., Scholl M. H. (2007b) OLAP Technology for Business Process Intelligence: Challenges and Solutions In: Data Warehousing and Knowledge Discovery: 9th International Conference, DaWaK 2007 Springer, pp. 111–122

Mayr H. C., Kop C., Esberger D. (2007) Business process modeling and requirements modeling. In: Digital Society, 2007. ICDS'07.. IEEE

Niedrite L., Solodovnikova D., Treimanis M., Niedritis A. (2007) Goal-driven Design of a Data Warehouse-based Business Process Analysis System. In: Proceedings 6th WSEAS Int. Conf. On Artificial Intelligence, Knowledge Engineering and Data Bases - Volume 6. World Scientific, Engineering Academy and Society (WSEAS), pp. 243–249

Pau K. C., Si Y. W., Dumas M. (2007) Data warehouse model for audit trail analysis in workflows. In: Student Workshop of the 2007 IEEE International Conference on e-Business Engineering (ICEBE 2007). IEEE

Shahzad K., Johannesson P. (2009) An evaluation of process warehousing approaches for business process analysis. In: Proceedings of the International Workshop on Enterprises & Organizational Modeling and Simulation. ACM

Shahzad K., Zdravkovic J. (2012) Process warehouses in practice: a goal-driven method for business process analysis. In: Journal of Software: Evolution and Process 24(3), pp. 321–339

Vogelgesang T., Appelrath H.-J. (2015) A Relational Data Warehouse for Multidimensional Process Mining. In: International Symposium on Data-Driven Process Discovery and Analysis. Springer, pp. 155–184

# OntoElecting Requirements for Domain Ontologies

## The Case of Time Domain

Vadim Ermolayev*,a

[a] Department of Computer Science, Zaporizhzhia National University, Ukraine.

Abstract. *This paper reports on the use of the OntoElect methodology for evaluating the fitness of an existing ontology to the requirements of the knowledge stakeholders in a domain. It demonstrates, that a thorough routine for indirect elicitation, ensuring completeness, correctness of interpretation, using in ontology evaluation of these requirements is a must for ontology engineering. This is also valid if the requirements for ontology refinement are elaborated by a very high profile expert working groups. The approach used in the reported research is based on the use of OntoElect – the methodology for ontology refinement. The workflow of OntoElect contains three phases: feature elicitation, requirements conceptualization, and ontology evaluation. It elicits the set of terms extracted from a saturated collection of documents in the domain. It further sublimates these terms to the set of required features using the information about term significance in the form of numeric scores. Furthermore, it applies conceptualization and formalization activities to these features yielding their aggregations as ontological fragments interpreted as formalized requirements. Finally, the mappings are specified between the elements in the requirements and ontology elements. The scores are used in the mappings to indicate the strength of positive or negative votes regarding the evaluated ontology. The sum of the votes gives the overall numeric fitness measure of the ontology to the domain requirements. The paper presents the use of OntoElect in the use case of evaluating the W3C OWL-Time ontology against the requirements extracted from the proceedings of the TIME symposia series.*

Keywords. Requirements Elicitation • Automated Term Extraction from Text • Feature Conceptualization • Ontology Engineering • Ontology Refinement • Ontology Evaluation • Ontology Fitness • Domain Knowledge Stakeholder • Vote • OntoElect

## 1 Introduction

Developing an ontology, in its lifecycle, with an aim to make it meet the requirements of the domain knowledge stakeholders is a complicated task. The State–of–the–Art approaches to ontology engineering still lack a rigorous engineering approach to measure the degree at which an ontology corresponds to the requirements. A major challenge is to elicit these requirements from the expert community around a domain in a way to ensure completeness and correct interpretation. Many popular methodologies, further mentioned in Section 2, suggest that the requirements have to be elicited via direct communication with the subject experts in the domain. Relying on this direct elicitation approach is however unrealistic. Subject experts quite often oppose direct approaches, like interviews, brainstorming sessions, etc., as an undesired overhead to their extensive commitments and prioritise these lowly. In interpreting requirements, the experts and knowledge engineers use different languages with various notations and expressive power. Thus, there is a need to either propose a lingua franca for both groups to share, or to find an indirect way to acquire domain understanding and requirements from the community of experts.

* Corresponding author.
E-mail. vadim@ermolayev.com

Shared languages are elaborated as simple, yet sufficiently expressive lexicons, that are constructed to be equally and easily interpreted by subject experts and ontology engineers in different domains Those could be positioned at a pre-design phase of the lifecycle of an information system or ontology. One notable framework is Klagenfurt Conceptual Pre-design Model (KCPM) by Kop et al. (2004). The alternative path is pursued within the field of Ontology Learning from Texts – see for example Wong et al. (2012) - which is a subfield in Text Mining. The most relevant indirect technique for requirement elicitation is, perhaps, Automated Term Extraction, which provides, together with the terms, the information to assess numerically their significance. However, the output of these extraction routines is just a flat set of terms labelling the required features. Consequently, the questions about the completeness of the requirements set and the correctness of their interpretations are left without an answer.

Even if one is lucky to have a complete and correct set of domain requirements shared by the community of subject experts, there is yet one more complication – the lack of objective quantitative metrics to assess the degree to which an ontology meets these requirements. This is not surprising because the representations of the requirements differ from ontology representations. The requirements to an ontology, to be correctly interpreted, have to be specified in a language which is easily understood by a domain expert who is not a professional ontologist. From the other hand, an ontology, to be properly used, has to be specified in a formal representation language, processable by machines, which reads weirdly to subject experts. Therefore, there is a definite need to have a way: (i) to transform requirements to a form, directly mappable to ontologies; and (ii) measure if the transformed requirements are fully implemented in an ontology.

On this way, a synergy between Conceptual Modelling and Ontology Engineering may help devise a proper engineering approach that answers the outlined questions and attempts to overcome the above mentioned challenging complications. This paper presents OntoElect – the methodology for ontology refinement that offers a rigorously measurable way to: (i) elicit and formalize the statistically representative sets of domain requirements; (ii) formalize these requirements as ontology fragments through conceptualization; and (iii) evaluate the domain ontology against these formalized requirements. To demonstrate that the methodology is valid, the use case in the domain of Time Representation and Reasoning is presented.

The remainder of the paper is structured as follows. Section 2 further explains the motivation toward developing a methodology such as OntoElect. It also briefly reviews the related work in light of seeking the answers to the important questions on assessing completeness and correctness of requirements, and also on evaluating the degree at which the ontology meets these requirements. Section 3 presents OntoElect in terms of describing its workflow, phases, techniques, formalisms, metrics and tools. Section 4 applies OntoElect to eliciting the requirements on time representation from the TIME collection of documents and evaluating the W3C OWL-Time ontology against these requirements. Section 5 summarizes the results and concludes the paper.

## 2 Motivation and Related Work

Roughly a decade ago, a reviewer of an ontology paper submitted to a Conceptual Modelling community conference remarked, that the paper would have been a good candidate to pass the test of time if the term Ontology had been replaced by the term Conceptual Model, also including related methodological issues. S/he also mentioned that ontolgies are a quick and careless way to sketch out conceptual models. Stimulated by this remark, I started thinking about how to further develop the craft of ontology design into real engineering. It turned out that both conceptual modelling and ontology development were crafts, at least regarding requirements elicitation. Both disciplines always claimed their careful attitude to the requirements of domain knowledge stakeholders. They did not however provide an objective and rigorous way

to measure if: (a) all significant requirements were put on the table; (b) all these requirements were correct or correctly interpreted; and (c) the final product met these significant requirements satisfactorily.

The mainstream in ontology engineering methodologies (Gómez-Pérez et al. (2004), Pinto et al. (2004), Schreiber (1999), Suárez-Figueroa et al. (2012), Sure et al. (2004) – to mention alphabetically the few most frequently cited) humbly forwards the answers to the questions on completeness, correctness, and fitness to requirements to the subject experts in the domain, through interviews, brainstorming sessions, or other ways of direct knowledge elicitation. The bottleneck is however that these experts often consider as inefficient the ratio of their own resource to be spent versus the utility of the resulting ontology as an artefact usable in their professional activity. Subject experts are sometimes also careful not to becoming less competitive and valuable to the company or community after making their knowledge available to the public in an explicit form. Therefore, indirect methods for requirements elicitation need to be developed to help provide a rigorous and explicit output in answering the above mentioned important questions.

## 2.1 Completeness

In Information Retrieval, completeness is often regarded as an indicator for results quality and expressed as recall metric. Recall is also used as one of the basic metrics for assessing quality in Automated Term Extraction which could be used as one of the enabling techniques for building terminology lists, thesauri etc. which could be considered as lightweight ontologies. More details on the related work in Automated Term Extraction, including approaches and implemented tools could be acquired from the review by Kosa et al. (2017).

Perhaps, one of the first mentions of the importance of a solution to the problem of completeness in Automated Term Extraction has been by Chien and Chen (2001) in the context of incremental term extraction from online text resources that are enlarged and extended over time. However, Chien and Chen (2001) looked at the problem from a linguistic perspective only and proposed a solution to analyse if all the term candidates have been extracted from a single textual document. OntoElect, proposed by Tatarintseva et al. (2013), looks at the problem broader and rather from statistical perspective. It suggests a method to measure the terminological completeness of the document collection by analyzing the saturation of terminological footprints of the incremental slices of the document collection, as for example reported by Ermolayev et al. (2014) regarding the domain of time representation and reasoning.

## 2.2 Correctness

The question on correctness reflects the long standing impedance mismatch between: (i) the requirements specified in a form clear to the domain knowledge stakeholders, but weird for the machine processing or for knowledge engineers; and (ii) ontologies specified in a way suited for machine processing but read weirdly by domain experts. Several approaches to resolve this mismatch could be found in the relevant literature.

One alternative, based on conceptual modelling and its lifecycle, is to propose a lingua franca – an easily understood subset of a conceptual modelling lexicon – that allows sharing and proper interpretation of requirement blueprints between subject experts and knowledge engineers. A notable representative of this approach is the Klagenfurt Conceptual Pre-design Model (KCPM) by Kop et al. (2004) developed in the NIBA project[1].

KCPM has been initially developed to bridge the above mentioned impedance mismatch, within the information system design cycle, between requirements specifications in natural (German) language and abstract conceptual models (for example, UML schemas). In KCPM, requirements are represented in a simplified yet formalized manner with a focus on the structural, functional, and behavioural terminology within an application domain. Regarding KCPM, there are at least two

[1] NIBA (Natürlichsprachige Informationsbedarfsanalyse) project has been funded by the Klaus Tschira Stiftung, Heidelberg.

aspects which are valuable for adoption, perhaps in an adapted form: (i) requirements are formalized to become closer to conceptual models; and (ii) requirements are focused on the terminology elaborated within the expert community in a domain. KCPM has been further developed by incorporating linguistic text processing in the requirements elicitation routine. Fliedl et al. (2005), based on KCPM and shallow text parsing, proposed an approach to bridge the application scenarios, taken in as natural language texts, and conceptual schemas describing the Universe of Discourse within a domain provided by the KCPM framework. These results inspired the development of the Conceptualization and Formalization pipeline in OntoElect presented in Section 3.3.

The approaches in ontology engineering based on the use of ontology design patterns, like those elaborated in frame of the NeOn EU project[2] and some other initiatives (e. g. (Presutti et al. 2012; Vrandecic 2010)] attack the correctness problem by offering reusable best practices in ontology engineering. They propose to shrink the space of opportunities for potential mistakes and mis-interpretations by offering ontology design blocks and patterns which were designed initially by renowned experts and passed the validity test. Further, design patterns may be effectively used for validating the correctness of an ontology, as proposed by Poveda-Villalón (2016). A disadvantage of this approach is that it is applied only to the output of an ontology engineering process, but not also to its input – to the requirements. Accordingly, it does help improve the quality of an ontology but does not help verify if the requirements were correctly met. Nevertheless, the results advancing this ontology engineering strand are quite useful in terms of re-using design patterns as higher-level elements for the specification and correct interpretation of requirements. These patterns are also useful in measuring structural and functional aspects of ontology quality, as proposed for example by Gangemi et al. (2006).

### 2.3 Ontology Fitness and Ontology Quality

Ontology quality is an issue which is broader than the focus of this paper. Here, one quality aspect – "'How well does an ontology meet the requirements?"' – is researched. It is however worth mentioning that there is a spectrum of aspects that need to be measured to assess the quality of an ontology.

Burton-Jones et al. (2005) proposed the suite of metrics to assess the quality of an arbitrary ontology, drawing upon semiotic theory. Their metrics assess the syntactic, semantic, pragmatic, and social aspects of quality. The metrics were operationalised and implemented in a prototype software tool called the Ontology Auditor.

A formal model for ontology evaluation and validation based on design patterns was proposed by Gangemi et al. (2006). Their model was based on the O2 meta-ontology and included three types of measures (structural, functional, and usability profiling). Based on this framework, they also elaborated the ontology of ontology validation called oQual. This quality evaluation framework was further extended and refined by Vrandecic (2010). Later, Poveda-Villalón (2016) proposed the set of patterns, metrics, and a tool for validating the correctness of an ontology as a design artefact.

It is known from the above mentioned literature, that one of the aggregate metrics for usability profiling of an ontology is fitness. Gangemi et al. (2006) distinguish fitness to competency questions and organizational fitness. Fitness to competency questions is in fact a way to assess how well an ontology meets the intended requirements, which is measured subjectively, as an opinion of a knowledge engineer and probably a subject expert. In difference to the predecessor work, OntoElect focuses explicitly on measuring fitness to requirements presented in a conceptualized and

[2] NeOn project has been co-funded by the European Commission's Sixth Framework Programme under grant number IST-2005-027595.

formalized manner. It suggests doing this by specifying mappings between ontological fragments and accounting for required feature significance – as described in Section 3.4.

## 3 An Outline of OntoElect

OntoElect seeks for maximizing the fitness of the developed ontology to what the domain knowledge stakeholders think about the domain. Fitness is measured as the ratio of stakeholders' positive over negative "votes" – a metric that allows assessing the stakeholders' commitment to the ontology under development – reflecting how well their sentiment about the requirements is met. The more positive votes are collected – the higher the commitment is expected to be. If a critical mass of votes is acquired (say 50%+1, which is a simple majority vote), the ontology is considered to satisfactorily meet the requirements. Votes, and information leading to quantifying votes are collected indirectly – extracted from a statistically representative document collection.

Hence, OntoElect is an ontology refinement methodology. It facilitates, in an unbiased and measured way, to find out what needs to be improved in the domain ontology to better meet the requirements of the domain knowledge stakeholders. OntoElect may also be used to cross-evaluate different ontologies, describing the same domain, by comparing their fitness measurements. Therefore, the inputs to OntoElect are (i) a carefully chosen collection of good quality documents which is deemed, by the knowledge stakeholders, to be representative of the domain; and (ii) an ontology describing this domain. The output of OntoElect is in fact the set of recommendations, based on measurements, on why this or that ontology describes the domain well or not very well and what needs to be improved in it.

### 3.1 OntoElect Workflow and Tools

The flow of activities in OntoElect is shown in Fig. 1. This workflow involves the two roles: knowledge engineers and subject experts. The major workload and coordination function falls on the knowledge engineers. The subject expert role involves those domain knowledge stakeholders who contributed their professional texts to the document collection describing the domain in question.The phases of the OntoElect workflow are:

- **Feature Elicitation.** Determine the saturated sub-collection of the chosen document collection representative of the domain. Within this sub-collection, determine the documents that provide the highest terminological impact on the domain – the decisive minority subset. Extract the set of multi-word terms from the decisive minority documents. Select the significant terms with their significance scores, further interpreted as required features.

- **Requirements Conceptualization and Formalization.** Categorize and group the required features, build feature taxonomy by elaborating subsumptions, part-whole relationships, feature inheritance, and memberships among the required features. Refine significance scores by accounting for their propagation through inheritance. Develop the feature taxonomy. Collect informal knowledge about the meaning of the required features and transform it to formalized structural contexts, further interpreted as requirements. Aggregate significance scores by giving account of the feature groupings in the requirements.

- **Ontology Evaluation.** Map the requirements to the appropriate ontology contexts. Compute positive and negative votes based on the: (i) similarities or dissimilarities revealed through context mappings; and (ii) aggregated significance scores. Compute the fitness of the ontology as the ratio of positive to negative votes. Make recommendations based on the votes for or against the most significant requirements.

The phases are further described in more detail.

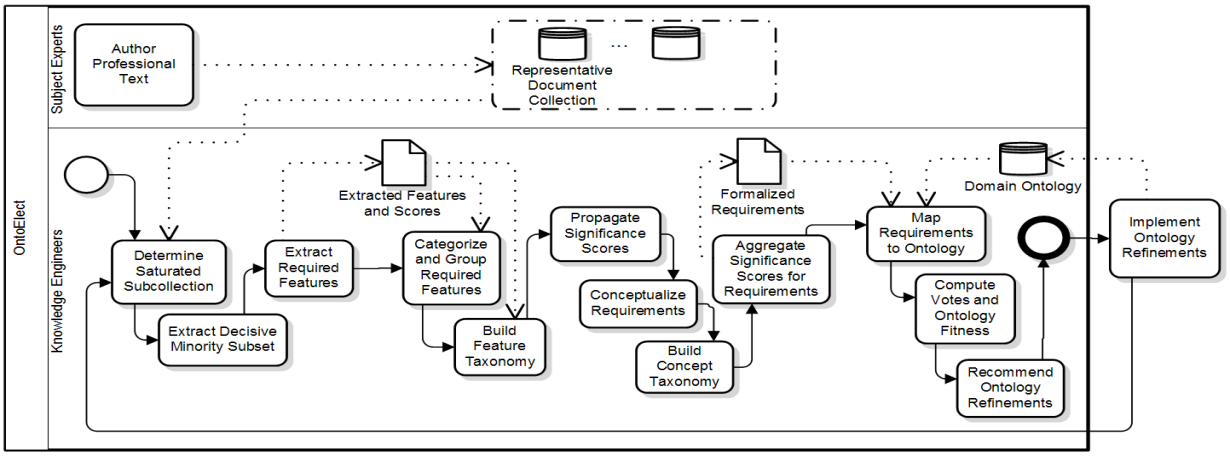



*Figure 1: OntoElect workflow, further elaborated from Tatarintseva et al. (2013). The phase of Implementing Ontology Refinements is pictured outside of the core workflows as it is beyond the scope of this paper*

### 3.2 Feature Elicitation Phase

As already mentioned in Section 2, direct acquisition of requirements from domain experts is not very realistic as they are expensive and not really willing to do the work falling out of their core activity. In OntoElect, we focus on the indirect collection of the stakeholders' votes by extracting these from high quality and reasonably high impact documents authored by the stakeholders in a domain.

An important feature to be ensured for knowledge extraction from text collections is that a collection needs to be statistically representative to cover the opinions of the domain knowledge stakeholders satisfactorily fully. OntoElect suggests a method to measure the terminological completeness of the document collection *by analyzing the saturation* of terminological footprints of the incremental slices of the document collection. The full texts of the documents from the collection are grouped in datasets in the order of their timestamps. As pictured in Fig. 2a, the first dataset $D1$ contains the first portion *(inc)* of documents. The second dataset $D2$ contains the first dataset $D1$ plus the second incremental slice *(inc)* of documents. Finally, the last dataset $Dn$ contains all the documents from the collection.

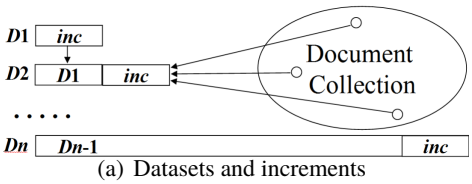
At the next step of the OntoElect workflow, the bags of multi-word terms $B1$, $B2$, ..., $Bn$ are extracted from the datasets $D1$, $D2$, ..., $Dn$, using UPM Term Extractor[3] software, together with their significance (*c-value*) scores. Those scores correlate to a significant extent to term frequencies – i.e. how often a term was met in the dataset. Please see an example of an extracted bag of terms in Fig. 2b.

At the subsequent step, every extracted bag of terms $Bi$, $i = 1$ , ..., $n$ is processed as follows:

- Individual term significance threshold *(eps)* is computed to cut off those terms that are not within the majority vote. The sum of *c-values* having values above *eps* form the majority vote if this sum is higher than *1/2* of the sum of all *c-values*.
- The cut-off at *c-value* < *eps* is done
- Normalized scores are computed for each individual term: *n-score = c-value / max(c-value)*
- The result is saved in $Ti$

After this step only significant terms, whose *n-scores* represent the majority vote, are retained in the bags of terms. $Ti$ are then evaluated for

[3] Java software for extracting terms and relations from scientific papers developed at Universidad Politechnica de Madrid in Dr Inventor EU project (https://github.com/ontologylearning-oeg/epnoi-legacy).




(a) Datasets and increments

| 1 | Term | c-value |
|---|---|---|
| 2 | temporal | 3453.000000 |
| 3 | time | 2287.500000 |
| 4 | interval | 1128.500000 |
| 5 | temporal logic | 1090.000000 |
| 6 | logic | 1049.000000 |
| 7 | constraints | 979.500000 |
| 8 | relations | 933.000000 |
| 9 | reasoning | 896.000000 |
| 10 | relation | 880.500000 |
| 11 | data | 815.500000 |
| 12 | intervals | 780.000000 |
| 13 | events | 773.500000 |
| 14 | temporal reasoning | 772.000000 |
| 15 | information | 766.000000 |
| 16 | value | 752.000000 |
| 17 | model | 721.000000 |
| 18 | temporal constraints | 694.000000 |
| 19 | temporal representation | 660.000000 |

(b) An example of an extracted bag of terms

*Figure 2: (a) Incrementally enlarged datasets in OntoElect; (b) an example of a bag of terms extracted by UPM Term Extractor*

saturation by measuring pair-wise terminological difference between the subsequent bags *Ti* and *Ti*+ 1 , $i = 0$ , ..., *n*-1. It is done by applying the THD algorithm by Tatarintseva et al. (2013). It is provided in Fig. 3 for reader convenience.

In fact, THD accumulates, in the *thd* value for the bag *Ti*+1, the *n-score* differences if there were linguistically the same terms in *Ti* and *Ti*+1. If there was not the same term in *Ti*, it adds the *n-score* of the orphan to the *thd* value of *Ti*+1. After *thd* has been computed, the relative terminological difference *thdr* receives its value as *thd* divided by the sum of *n-scores* in *Ti*+1.

Absolute *(thd)* and relative *(thdr)* terminological differences are computed for further assessing if *Ti*+1 differs from *Ti* more than the individual term significance threshold *eps*. If not, it implies that adding an increment of documents to *Di* for producing *Di*+1 did not contribute any noticeable amount of new terminology. Thus, the subset *Di*+1 of the overall document collection may have become terminologically saturated. However, to obtain more confidence about the saturation, OntoElect suggests that some more subsequent pairs of *Ti* and *Ti*+1 are evaluated. If stable saturation is observed, then the process of looking for a minimal saturated sub-collection could be stopped. Sometimes, however, a terminological peak may occur after saturation has been observed in the previous pairs of *T*. Normally, this peak indicates that a highly innovative document with a substantial number of new terms has been added in the increment.

Additionally, the documents in the saturated sub-collection which have the major terminological impact on domain coverage are found out. As reported in Ermolayev et al. (2014), those are the most frequently cited documents. The numbers of citations for each paper are acquired from Google Scholar[4] using the Catalogue Generator software tool (Kosa et al. 2017). Citation frequencies *cfr* (the number of citations per year) are computed, and the impact of each paper in the collection is computed as:

$$imp = \begin{cases} [0.2 \times cfr] + 1, cfr > 0 \\ 0, cfr = 0 \end{cases} \quad (1)$$

where the square brackets stand for taking integer part. Hence, the contribution of the frequency of citations to the impact of the paper is weighted by 0.2, while the papers having no citations are filtered out. The documents having their impact value *imp* > *threshold* form the decisive

[4] http://scholar.google.com/

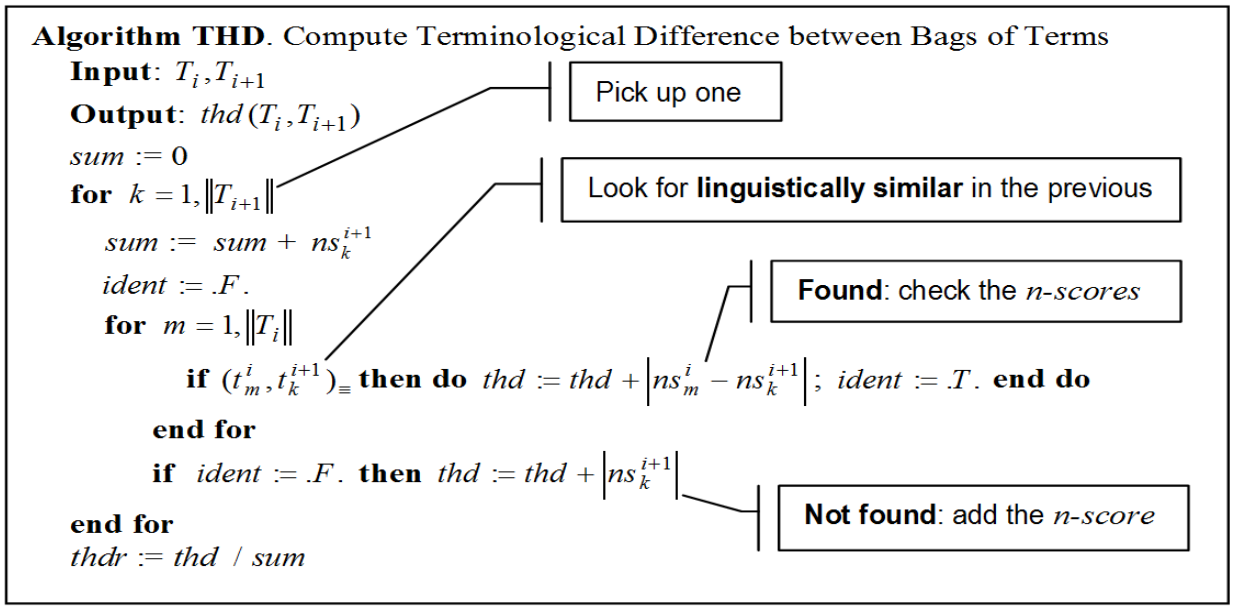



*Figure 3: THD algorithm for computing terminological difference in a pair of bags of terms*

minority sub-collection of the entire document collection. The *threshold* for filtering out documents from the decisive minority sub-collection is chosen empirically to ensure that the majority of the most significant terms are retained.

The Feature Elicitation Phase of OntoElect outputs the ranked list of the terms which represent the stakeholder sentiment about the domain. This list is extracted from the decisive minority sub-collection of the documents. Finally, the list is examined by a knowledge engineer and the irrelevant terms are withdrawn. These irrelevant terms may appear in the list because these may frequently appear in the collection and are often individuals in terms of ontology structure. For example, if the collection of conference proceedings is used for term extraction, the names of the authors of the frequently cited papers, affiliations, venues, names of the popular datasets used in experiments, etc. may receive high scores and be considered as significant. Among those mentioned above, only the names of the datasets may be somehow relevant for an ontology. The rest have to be withdrawn.

The terms in the cleaned list are further interpreted as the required features to be met by the ontology under evaluation or refinement.

To finalize this brief presentation of the OntoElect feature elicitation phase, it is worth noting that it is domain independent and unsupervised. However, the particular term extraction tool implies that it is able to process only English documents. To compensate this shortcoming, the processing pipeline is architectured in a modular fashion. Thus, it is possible to replace the term extraction tool by another one, for example for a different language.

### 3.3 Requirements Conceptualization and Formalization Phase

The task for this phase of OntoElect is to transform the ranked list of the required features to formalized ontological fragments (requirements), carrying the positive and negative votes of the involved features in their aggregated significance scores. Requirements are further used in Ontology Evaluation phase to compute the fitness of the ontology. Conceptualization and formalization in OntoElect are done by a knowledge engineer through:

- Grouping and categorizing extracted required features. Individual features could be interpreted as concepts, properties, or individuals.

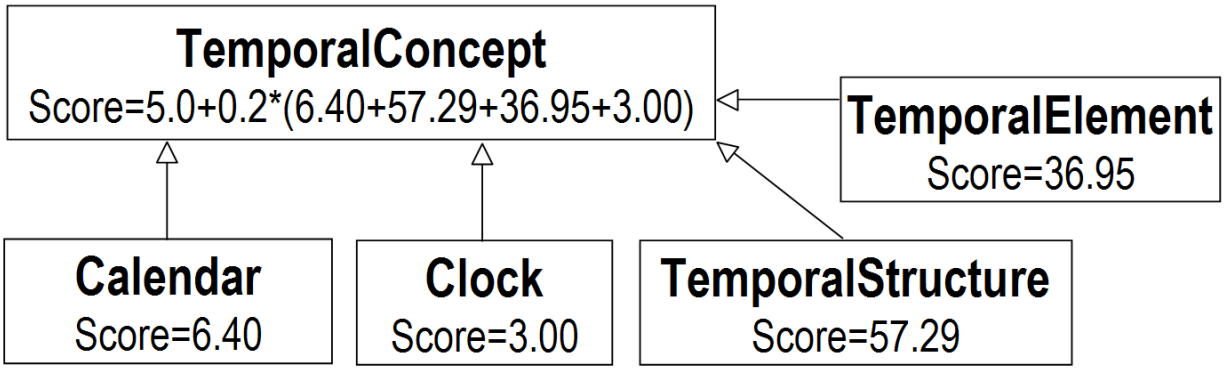


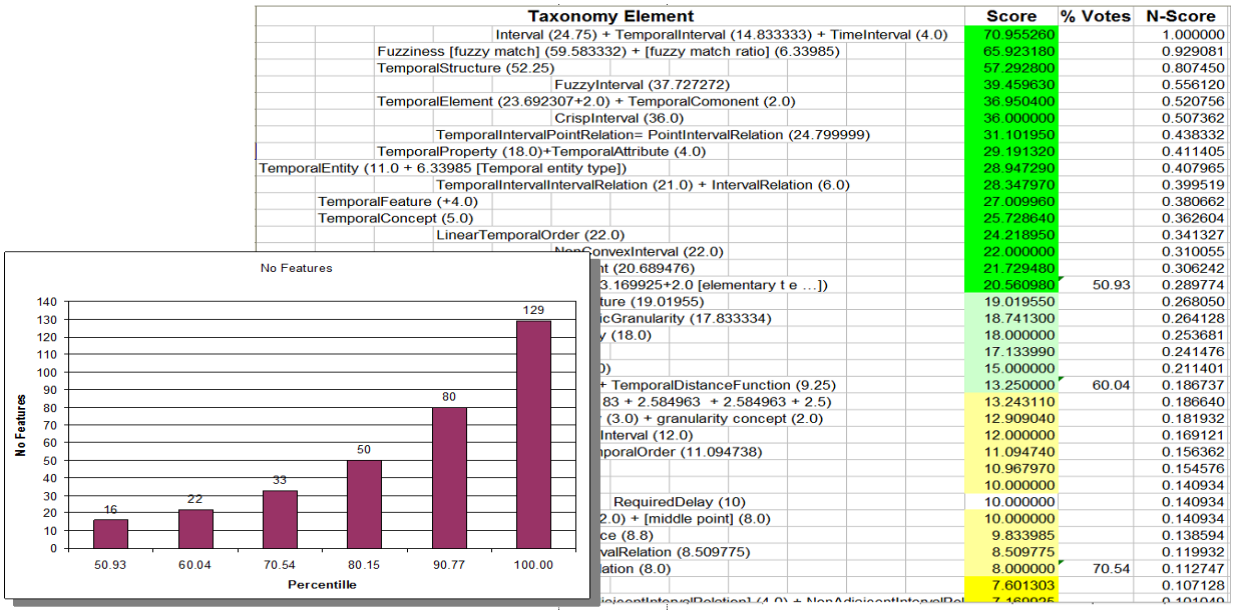

*Figure 4: An example of computing score propagation for required features*

- Selecting the significant concepts from the list of required features and forming the feature taxonomy, also including property features in the appropriate concept features
- Computing the propagated scores up the concept/property hierarchies
- Selecting the most significant concept features
- Elaborating natural language definitions for the most significant concept features and formalizing these as ontological fragments using UML and OWL
- Documenting requirements

Feature grouping is merging several features which are lexically different but carry equivalent semantics. The relevant cases include: plural and singular forms of the same term, for example "temporal constraints" and "temporal constraint" are the same terms and have to be merged; the terms that had or had not lost two-letter combinations because of the peculiarities of their representation in PDF documents due to the differences in Adobe versions, for example "de nition" and "definition" are also the same terms. The significance scores of the merged terms are added.

Feature categorization stands for deciding if a feature, due to its semantics, represents a concept, a property, or an individual. Concept features are further used to form subsumption or meronymy hierarchies in the feature taxonomy. The root of the feature taxonomy is the most abstract and general "thing" concept to which the rest of the concept features directly or indirectly subsume. For example, a temporal interval subsumes to temporal thing, etc. Meronymy hierarchies involve concept features which are either parts of a whole, like a weekend is the part of a week, or the wholes for their parts. Both types of these hierarchical relationships are important as they influence the significance of features through property inheritance. Indeed, if a feature subsumes to another feature then it inherits some of its properties – so its significance is formed to a particular extent by these inherited properties. Hence, a parent in a hierarchy may expect that it is rewarded by its children through the propagation of their significance scores. OntoElect suggests (Tatarintseva et al. 2013) that score propagation adds one fifth of the children' scores to their parent's score. An example of computing propagated scores is pictured in Fig. 4.

After propagating the scores in the feature taxonomy, the most important concepts in it, having the potential for high impact on the requirements due to their scores, may be selected. For that, concept features are viewed in a ranked list and the group of features covering the desired proportion of importance is promoted. An example of several (percentile) groups of concept features for the time domain is given in Fig. 5. The promoted concept features are used to form the concept taxonomy and be the central concepts




| Taxonomy Element | Score | % Votes | N-Score |
|---|---|---|---|
| Interval (24.75) + TemporalInterval (14.833333) + TimeInterval (4.0) | 70.955260 | | 1.000000 |
| Fuzziness [fuzzy match] (59.583332) + [fuzzy match ratio] (6.33985) | 65.923180 | | 0.929081 |
| TemporalStructure (52.25) | 57.292800 | | 0.807450 |
| FuzzyInterval (37.727272) | 39.459630 | | 0.556120 |
| TemporalElement (23.692307+2.0) + TemporalComonent (2.0) | 36.950400 | | 0.520756 |
| CrispInterval (36.0) | 36.000000 | | 0.507362 |
| TemporalIntervalPointRelation= PointIntervalRelation (24.799999) | 31.101950 | | 0.438332 |
| TemporalProperty (18.0)+TemporalAttribute (4.0) | 29.191320 | | 0.411405 |
| TemporalEntity (11.0 + 6.33985 [Temporal entity type]) | 28.947290 | | 0.407965 |
| TemporalIntervalIntervalRelation (21.0) + IntervalRelation (6.0) | 28.347970 | | 0.399519 |
| TemporalFeature (+4.0) | 27.009960 | | 0.380662 |
| TemporalConcept (5.0) | 25.728640 | | 0.362604 |
| LinearTemporalOrder (22.0) | 24.218950 | | 0.341327 |
| NonConvexInterval (22.0) | 22.000000 | | 0.310055 |
| nt (20.689476) | 21.729480 | | 0.306242 |
| 3.169925+2.0 [elementary t e ...]) | 20.560980 | 50.93 | 0.289774 |
| ture (19.01955) | 19.019550 | | 0.268050 |
| icGranularity (17.833334) | 18.741300 | | 0.264128 |
| y (18.0) | 18.000000 | | 0.253681 |
| | 17.133990 | | 0.241476 |
| ) | 15.000000 | | 0.211401 |
| + TemporalDistanceFunction (9.25) | 13.250000 | 60.04 | 0.186737 |
| 83 + 2.584963 + 2.584963 + 2.5) | 13.243110 | | 0.186640 |
| (3.0) + granularity concept (2.0) | 12.909040 | | 0.181932 |
| Interval (12.0) | 12.000000 | | 0.169121 |
| poralOrder (11.094738) | 11.094740 | | 0.156362 |
| | 10.967970 | | 0.154576 |
| | 10.000000 | | 0.140934 |
| RequiredDelay (10) | 10.000000 | | 0.140934 |
| 2.0) + [middle point] (8.0) | 10.000000 | | 0.140934 |
| ce (8.8) | 9.833985 | | 0.138594 |
| valRelation (8.509775) | 8.509775 | | 0.119932 |
| ation (8.0) | 8.000000 | 70.54 | 0.112747 |
| | 7.601303 | | 0.107128 |

*Figure 5: An example of the groups of concept features per their importance. The upper group in the list contains 16 features which cover 50+0.93 percent of the accumulated significance (scores) of all the 129 retained features. The diagram to the left pictures the distribution of the features per percentile groups.*

for the formalized requirements. Each of these promoted concept features is conceptualized in a formalized ontological fragment – as a conceptual model and a piece of code in an ontology specification language. Conceptualization means that all the relevant property features and features representing individuals are consolidated in the ontological fragment in a harmonized way to form a coherent piece of a required descriptive theory for the domain.

The subsequent step in conceptualizing a concept feature is elaborating its natural language definition based on the documents in the collection and probably some external resources of high reputation. The task of a knowledge engineer within this step is to ensure that all the required property features are taken into this definition and do not contradict each other. Based on the natural language definition, a conceptual model is developed for this concept feature, including also its properties and relationships to the other relevant concept features.

OntoElect does not enforce any specific working pattern or software tool for a knowledge engineer at this formalization step. In our development practices formalization is a two step process. The first step is updating the conceptual model coded as a UML 1.4 class diagram (Booch et al. 2000) using the ArgoUML editor[5]. Protege ontology editor[6] is used in the second step for coding the ontology in OWL 2 with an account for DL restrictions (Motik et al. 2012). The transformation patterns from UML to OWL follow the recommendations by Schreiber[7]

OntoElect is more specific in recommending a way for documenting the ontology under development. It suggests that the ontology is documented in a set of Semantic MediaWiki[8] pages. Some of those pages provide the overviews of the ontology modules, but the rest, which are the majority, are dedicated to documenting the concepts – one page

[5] ArgoUML is an open source UML modeling tool:http://argouml.tigris.org/

[6] Protege Ontology Editor: https://protege.stanford.edu/

[7] OWL Restrictions: http://www.cs.vu.nl/~guus/public/owl-restrictions/

[8] http://semantic-mediawiki.org/

per concept. A documentation wiki page of a particular concept contains:

- The natural language definition of the concept
- The UML class diagram of the concept's conceptual model
- The description of the concept's properties grouped according to the property types: datatype and object properties

### 3.4 Evaluation Phase

The objective of this phase is to figure out how well does an ontology ($O$) describe the domain meets the formalized requirements ($R$). This is done by mapping the requirements, as ontological fragments represented by their central concept features, to the semantically corresponding structural contexts within the ontology. The mappings reveal either similarity or dissimilarity and, therefore, either increase or decrease the fitness of $O$. To explain this with a little bit of rigor and present a way to visualize ontology fitness to domain requirements, an allusion of a gravitation field, proposed by Ermolayev (2015), will be further used.

Let us assume that a domain ($D$) is adequately modelled by the set of all relevant requirements ($R$). For building a grid based on these requirements it is assumed, as pictured in Fig. 6a, that:

- All the requirements are placed in the centre of $D$; and
- They are not equal in their significance – i.e. have different spheres of influence around the centre of gravitation, which is quantified using the normalized significance scores $ns \in [0, 1]$

Let us suppose now that an ontology ($O$) is positioned in $D$ at a distance l from its centre (Fig. 6(b)). This can be any location on the circle of radius l around the centre of the grid (Fig. 6(a)). Let us now reveal what might be the forces influencing $O$ in this position.

Let us assume that $O$ is checked against the requirements $r$ from $R$ which spheres of influence reach the position of $O$ *(i.e.* $ns_r \geq l$*)*. The following are the possible outcomes of these checks:

- A particular part of $O$, say a semantic context $o \in O$ (a white coloured circle in Fig. 6(b)), meets the requirement $r$. Therefore, $O$ becomes more fitting to $R$. In this case we will consider that the increase in fitness ( $\Delta\Phi_0^+$) creates a positive gravitation force $\overrightarrow{G}_o^+$ applied to $O$ and directed towards the centre of $D$, as pictured in Fig. 6(b). The absolute value of this force is computed using a direct analogy with the Law of Universal Gravitation by Newton (1999):

$$G_o^+ = \frac{1 \times \Delta\Phi_o^+}{(ns_r)^2}, \quad (2)$$

  where: "1" in the numerator is the fitness of $r$ with respect to $D$ – meaning that $r$ fits $D$ perfectly as one of its requirements; the value of $\Delta\Phi_0^+$ is within [0,1].

- There is no semantic context $o \in O$ that meets the requirement $r$ (no circle on the ontology side in Fig. 1(b)) or there is an $o$ that contradicts $r$ (a black coloured circle in Fig. 6(b)). In both cases $O$ becomes less fitting to $R$. Therefore, we will consider that the decrease in fitness ( $\Delta\Phi_O^-$ for a missing semantic context; $\Delta\Phi_o^-$ for a context contradictory to $r$) creates a negative gravitation force, $\overrightarrow{G}_O^-$ or $\overrightarrow{G}_o^-$ applied to $O$ and directed towards the periphery of $D$, as pictured in Fig. 6(b). Similarly to (2), the absolute values of these forces are computed as:

$$G_O^- = \frac{1 \times \Delta\Phi_O^-}{(ns_r)^2}, G_o^- = \frac{1 \times \Delta\Phi_o^-}{(ns_r)^2} \quad (3)$$

The overall gravitation force applied to $O$ as an influence by $D$ is computed as a vector sum:

$$\overrightarrow{G}_O\big|_D = \sum_{r \in R: ns_r \geq l} \left(\overrightarrow{G}_o^+ + \overrightarrow{G}_O^+ + \overrightarrow{G}_o^-\right) \quad (4)$$

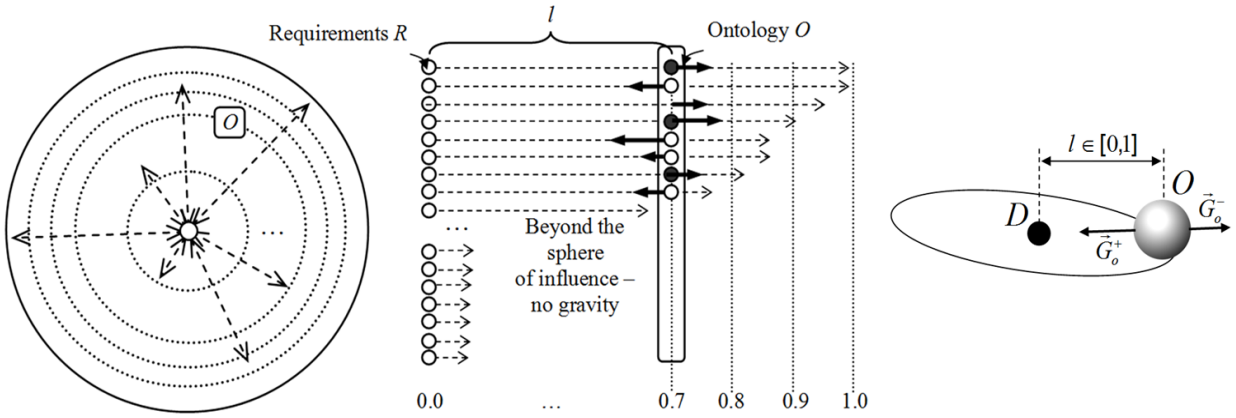



*Figure 6: Domain requirements: (a) their spheres of influence; (b) gravitation forces; and (c) the equilibrium state of ontology O in D – adopted from Ermolayev (2015)*

$O$ is considered as properly positioned within $D$ when it reaches its *equilibrium state* (Fig. 6(c)) with respect to the gravitation field in $D$, i.e. appears at a distance l from the centre of $D$ at which $\overrightarrow{G_O}\big|_D = \overrightarrow{0}$. This distance could be interpreted as an integral measure of the semantic difference between what does $O$ describe and what is required to be described for $D$ by its knowledge stakeholders. If $O$ is not in an equilibrium state regarding $D$, $\overrightarrow{G_O}\big|_D$ will cause it to move either towards the centre of $D$ or towards its periphery.

Equivalence mappings are created to measure, based on significance scores, the degree of (dis-)similarity at the schema level between the required features and the elements in the ontology under evaluation. A mapping is a relationship between a concept feature and a concept in the ontology, or a property feature and the property in the ontology:

$$\mu(f, e) = \langle \equiv; l; e; r; score; n - score[; cf] \rangle, \quad (5)$$

where: $\equiv$ is the signature (equivalence mapping); $f$ is the required feature; $e$ is the corresponding ontology element; $r$ is the ratio of similarity of $f$ to $e$; *score* is the aggregated score of $f$; *n-score* is its normalized aggregated score to determine its sphere of influence; and $cf$ is an optional confidence factor provided by the knowledge engineer and equal to 1 by default.

Each mapping (5) is a way to specify a positive vote in the sense of (2) or a negative vote in the sense of (3-4). Thus, the task of a knowledge engineer at this OntoElect phase is, for every formalized requirement, to specify the set of mappings of the features aggregated in this requirement to the elements of the evaluated ontology. When done, (s)he may compute the values of positive and negative votes and, further the value for the overall ontology fitness.

## 4 Evaluating OWL-Time against Time Domain Requirements

The progress in understanding the World and its data in their dynamics is based on having an adequately expressive model of time and, therefore, pushes forward the refinement of time models. The developments in Philosophy, Artificial Intelligence, Databases, Distributed Systems, etc. in the last two decades have brought to life several prominent theoretical frameworks dealing with temporal aspects. Some parts of these theories gave boost to research in logics – yielding a family of temporal logics comprising temporal description logics. Based on this foundation, knowledge representation languages have received their capability to represent time, and several ontologies of time have been implemented by the Semantic Web community. It is however important to find

out if this plenty is enough to meet the demand in Semantic Data Management.

### 4.1 The Use Case of OWL-Time

One of the most widely used temporal ontologies is W3C OWL-Time initially developed by Cox et al. (2006) as W3C Working Draft dated 27 September 2006. Since that the ontology has been stalled as recognized by the Consortium. In its current shape, however, OWL-Time has noticeable shortcomings. For example, as articulated by the experts in the W3C Spatial Data on the Web Working Group (SDW WG)[9], "... one of the shortcomings of OWL-Time is that it is unclear how to use OWL-Time in practice, especially how you query temporal data in ISO 8601 via OWL-Time." The members of the SDW WG committed to deliver a refined ontology, based on OWL-Time, in a year time frame.

The WG members, based on their expertise, also articulated and discussed the important requirements to this refined ontology. Among these requirements mentioned by the experts were:

- Non-Gregorian calendars
- Other (than currently in OWL-Time) time (TimeStamp) formats
- Approximate time instants (TimeStamp)
- Periods like Cretaceous period
- Leap seconds

Valid questions regarding this use case for OntoElect is if these requirements articulated and discussed by the SDW WG members are complete, accurate, and important.

As it is demonstrated in the subsequent sections, OntoElect allows answering these questions. Completeness is checked by comparing the list to the requirements elicited and conceptualized from the TIME paper collection presented in Section 4.2. The result of extracting required features from this document collection is presented in Section 4.3.

The TIME community has been chosen for the use case as the members of the SDW WG outlined the need to query temporal data via an ontology. Consequently, Time Representation and Reasoning looks like a very relevant community.

### 4.2 TIME Document Collection and Datasets

To assess the sufficiency of domain coverage, the consensual set of the features of time has to be extracted and appropriately structured. A way to do that is to analyze the document corpus produced by the appropriately chosen professional community and extract the required features from there – the TIME community (http://time.di.unimi.it/) in this case. The document corpus for required features extraction has been formed of the proceedings papers of the TIME Symposia series published by IEEE. The collection contained all the papers published in the TIME symposia proceedings between 1994 and 2013, which are 437 full text documents in total. The papers of this collection have been pre-processed manually, including their conversion to plain texts and cleaning of these texts. Accordingly, the resulting datasets were not very noisy. The datasets have been generated using Dataset Generator[10] module from the OntoElect Instrumental Toolset by Kosa et al. (2017). We have chosen the increment for generating the datasets to be 20 papers. Moreover, based on the available texts, we have generated 22 incrementally enlarged datasets $D1, D2, ..., D22$[11]. For generating the datasets the chronological order of adding documents has been used.

[9] Here and below in this section, we cite and use the facts from the minutes of the W3C SDW WG meeting on the 09 February 2016. The document is available at https://www.w3.org/2016/02/08-sdw-minutes#item07.

[10] The Dataset Generator is available at: https://github.com/bwtgroup/SSRTDC-PDF2TXT. More details, also on the other software modules of the OntoElect Instrumental Toolset, are given in Kosa et al. (2017).

[11] The TIME collection in plain text and the datasets generated of these texts are available at: https://www.dropbox.com/sh/64pbodb2dmpndcy/AAAzVW7aEpgW-JrXHaCEqg2Sa/TIME?dl=0.

### 4.3 Feature Elicitation

For extracting terms from TIME datasets, the UPM Term Extractor software has been deliberately chosen by Kosa et al. (2017) as an appropriate tool for automated term extraction from plain text with respect to measuring terminological saturation. The results of measuring terminological difference and detecting terminological saturation are presented in Table 1 and pictured in Fig. 7.

The saturation measurements **revealed stable saturation** starting from $D11 - D12$ – as presented in Table 1 by bold values and pictured in Fig. 7 by the vertical dashed line. The saturation curve has terminological peaks hinting about the appearance of documents with higher terminological contributions. Saturation is detected at *eps* equal to 23.774. The number of retained terms in $T12$ is 7110, which is only 2.47% of the total number of extracted terms in the corresponding bag of terms $B12$.

Following Ermolayev et al. (2014), we selected the decisive minority sub-collection using the information about the frequency of citations, as described in Section 3. The paper terminological impact *threshold* has been chosen as equal to 2. The decisive minority sub-collection contained 24 papers. A single dataset was formed of these 24 papers and terms extracted using UPM Term Extractor. The resulting bag of terms contained 686 terms after withdrawing the irrelevant entries and grouping.

### 4.4 Conceptualizing and Formalizing Requirements

For easier cleaning, the ranked list of retained terms has been classified as indicated in Table2[12]. Each term has been put into only one category.

This classification helped withdraw the terms attributed to the groups deemed as not fully relevant – all except Features. Furthermore, cleaning reduced the set of feature candidates to 175 items. The terms in the candidate list have been grouped – yielding 129 features.

The results of grouping are pictured in Fig. 5. For instance, it shows in the first row of the table that the feature of TimeInterval has been grouped by merging the features of Interval (with significance score of 24.75), TemporalInterval (14.83), and TimeInterval (4.0). Moreover, the significance score of a TimeInterval became 43.58. Significance score propagation has then been done for all 129 required features. Fig. 5 shows for example that the TimeInterval feature has received the additions in its score at least from FuzzyInterval (39.45), CrispInterval (36.0), and NonConvexInterval (22.0)[13]. After adding the propagated scores, the significance of TimeInterval became equal to 70.96, which made it the top scoring required feature.

As it may also be seen in Fig. 5, some features, like TimeInterval or TemporalStructure, could be categorized as concept features. Many other features, for example TemporalIntervalRelation or Fuzziness, read as properties and were categorized as property features. There were also a few features that read as individuals, those however were quite modestly scored in significance and were further neglected as not really important.

The taxonomy of temporal features has been further developed as shown in Fig. 8. The authenticity of the names was preserved from the above mentioned list (Fig. 5) to a maximal extent.

Already at this stage, it is possible to check if the requirements for refining OWL-Time discussed at the W3C SDW WG meeting (Section 4.1) are significant compared to the sentiment of the TIME community. The correspondences are presented in Table 3.

The analysis of Table 3 reveals that the requirements discussed by the experts in the SDW WG do not correspond to the most significant required features. The reason might be that these top ranking required features were already properly implemented in OWL-Time. This hypothesis will

[12] The complete table may be accessed at http://ermolayev.com/TimeOnto/ClassifiedTerms.zip.

[13] The other subsumed features are not visible in Fig. 5 – for example InfiniteInterval (12.0)

*Table 1: Saturation measurements for the TIME bags of terms extracted by UPM Term Extractor*

| Datasets Pair | No of Terms | | Cut-off threshold *eps* | Retained Terms *(c-value >eps)* | *thd*, value | *thdr,%* |
|---|---|---|---|---|---|---|
| | in the Bag of Terms *Bi* | With *c-value* >1 | | | | |
| empty-D1 | 53478 | 13775 | 28.000000 | 1379 | 112.240776 | 100.000000 |
| D1-D2 | 91701 | 23816 | 24.000000 | 2473 | 72.425797 | 59.389624 |
| D2-D3 | 114061 | 32419 | 21.500000 | 3028 | 24.265441 | 17.312132 |
| D3-D4 | 129896 | 39643 | 19.651484 | 3997 | 32.879384 | 20.295700 |
| D4-D5 | 145796 | 46702 | 19.651484 | 4466 | 32.622249 | 17.809632 |
| D5-D6 | 162746 | 54629 | 20.000000 | 4587 | 44.646245 | 27.027091 |
| D6-D7 | 190263 | 63684 | 21.000000 | 5133 | 38.071510 | 24.076680 |
| D7-D8 | 200176 | 69097 | 22.000000 | 5413 | 26.869088 | 18.598430 |
| D8-D9 | 217461 | 76315 | 22.000000 | 5855 | 18.776156 | 13.110501 |
| D9-D10 | 245967 | 84664 | 23.219281 | 6453 | 26.914239 | 18.281013 |
| D10-D11 | 263034 | 91132 | 24.000000 | 6428 | 24.164533 | 16.688847 |
| D11-D12 | 287887 | 99231 | **23.774438** | **7110** | **18.109566** | **12.737127** |
| D12-D13 | 298367 | 104398 | 23.774438 | 7383 | 12.573733 | 9.144105 |
| D13-D14 | 320500 | 112898 | 24.000000 | 7723 | 13.334954 | 9.624406 |
| D14-D15 | 333975 | 119787 | 23.774438 | 8298 | 14.403930 | 10.698614 |
| D15-D16 | 350741 | 127257 | 24.000000 | 8426 | 16.428110 | 13.135633 |
| D16-D17 | 369316 | 135085 | 24.000000 | 8877 | 9.642629 | 7.638542 |
| D17-D18 | 389022 | 143452 | 24.000000 | 9617 | 11.416546 | 8.784302 |
| D18-D19 | 399553 | 148896 | 24.000000 | 10005 | 8.042102 | 6.136623 |
| D19-D20 | 420464 | 158179 | 24.000000 | 10574 | 11.655716 | 8.652365 |
| D20-D21 | 435075 | 165519 | 26.000000 | 9751 | 9.781677 | 7.297311 |
| D21-D22 | 449719 | 171135 | 26.000000 | 10139 | 6.926144 | 5.109224 |

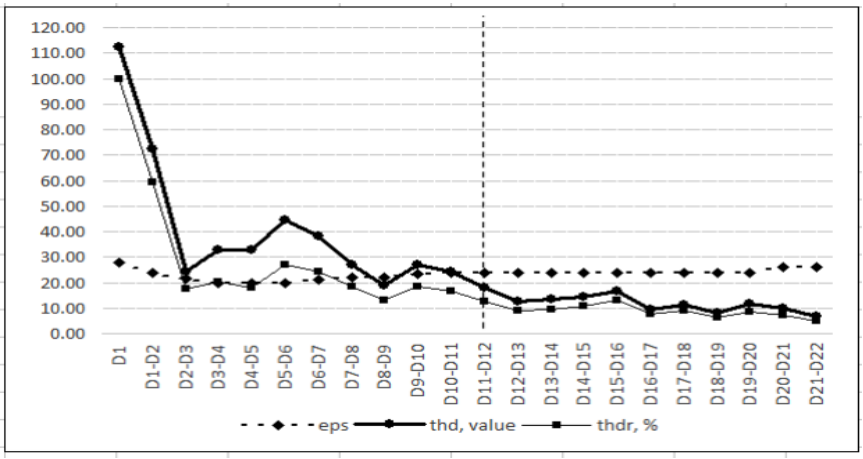
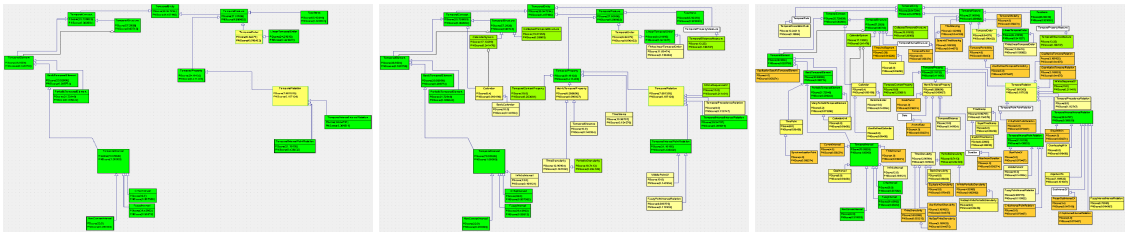



*Figure 7: Saturation measurements on the TIME datasets based on the bags of terms extracted using UPM Term Extractor*

*Table 2: The first fourteen terms extracted from the decisive minority sub-collection dataset and their classification. The numbers under the categories indicate the total quantities of the terms under each category.*

| Score | Term | **Logic** | **Problem** | **Formula** | **Formalism** | **Operator** | **Method** | **Model** | **Reasoner** | **Domain** | **Language** | **Feature** | **Constraint** | **Instance** | **Pattern** | **Application** | **Project** | **Author** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Total No of terms: **686** | 44 | 27 | 6 | 36 | 8 | 22 | 24 | 1 | 4 | 8 | 175 | 28 | 1 | 13 | 110 | 1 | 178 |
| 147.11 | temporal logic | ✓ | | | | | | | | | | | | | | | | |
| 100.11 | calendar pattern | | | | | | | | | | | | | | ✓ | | | |
| 86.54 | temporal constraint | | | | | | | | | | | | ✓ | | | | | |
| 68.73 | temporal operator | | | | | ✓ | | | | | | | | | | | | |
| 59.58 | fuzzy match | | | | | | | | | | | ✓ | | | | | | |
| 52.25 | temporal structure | | | | | | | | | | | ✓ | | | | | | |
| 49.83 | calendar schema | | | | | | | | | | | ✓ | | | | | | |
| 46.25 | temporal representation | | | | ✓ | | | | | | | | | | | | | |
| 41.00 | temporal reasoning | | | | | | ✓ | | | | | | | | | | | |
| 40.00 | freeze quantifier | | | | ✓ | | | | | | | | | | | | | |
| 37.73 | fuzzy interval | | | | | | | | | | | ✓ | | | | | | |
| 36.36 | xml document | | | | | | | | | | | | | | | ✓ | | |
| 36.00 | crisp interval | | | | | | | | | | | ✓ | | | | | | |
| 34.00 | satisfiability problem | | ✓ | | | | | | | | | | | | | | | |



(a) 50 percentile – 16 features (b) 70 percentile – 33 features (c) 90 percentile – 80 features

*Figure 8: Required feature taxonomy. The numbers of features in percentiles (a) – (c) and the colours of the included required features correspond to the diagram in Fig. 5.*

*Table 3: The correspondences between the required features extracted using OntoElect and requirements by W3C SDW WG*

| Required Features Extracted from TIME | Signi-ficance | W3C SDW WG Requirements |
|---|---|---|
| TimeInterval | 70.96 | |
| Fuzziness | 65.92 | |
| TemporalStructure | 57.29 | |
| FuzzyInterval | 39.46 | |
| TemporalElement | 36.95 | |
| CrispInterval | 36.0 | |
| PointIntervalRelation | 31.10 | |
| TemporalProperty | 29.19 | |
| TemporalEntity | 28.95 | |
| IntervalIntervalRelation | 28.35 | |
| TemporalFeature (high-level) | 27.01 | Approximate time instants |
| TemporalConcept | 25.73 | |
| LinearTemporalOrder | 24.22 | |
| NonConvexInterval | 22.0 | |
| PeriodicTemporalElement | 21.73 | Periods like Cretaceous period |
| BasicTemporalElement | 20.56 | |
| ... | | |
| TimeStamp | 10.97 | Other (than currently in OWL-Time) time (TimeStamp) formats |
| ... | | |
| Calendar | 6.4 | Non-Gregorian calendars |
| ... | | |
| Clock | 3.0 | Leap seconds |
| ... | | |

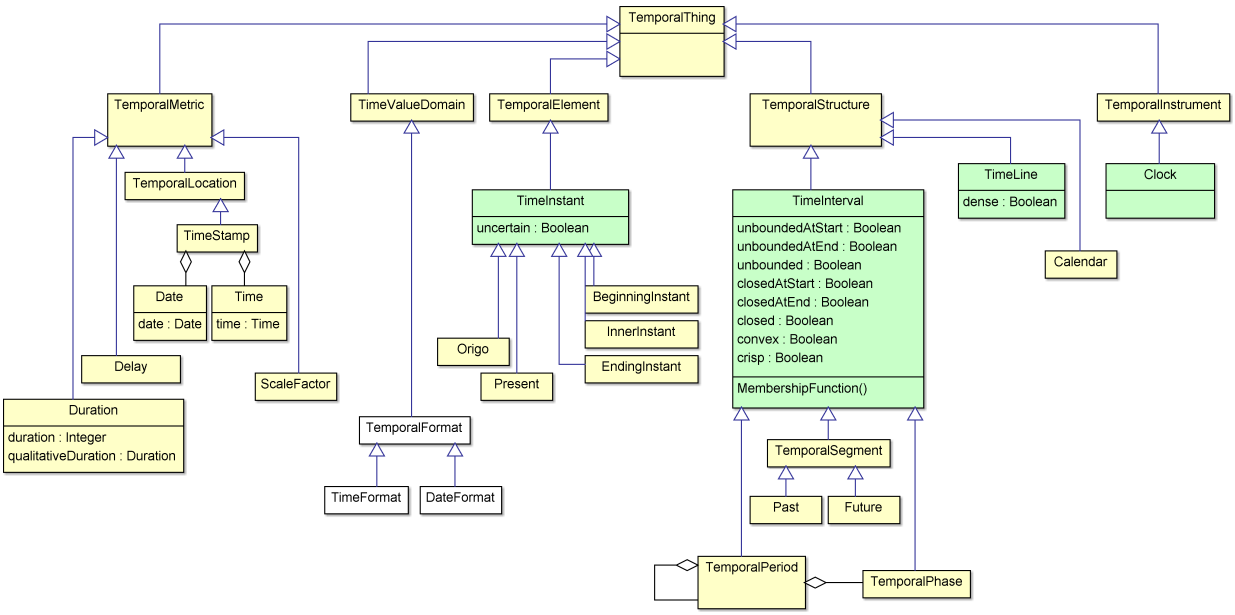



*Figure 9: The concept taxonomy of TIME requirements. Significant concept features, that constitute, together with their subsumed hierarchies and aggregated property features, the 70 percentile (Fig. 8(b)), are coloured green.*

be checked in Section 4.5 by building the equivalence mappings between the ontological fragments of OWL-Time and formalized requirements.

The hierarchy of the concept features presented in a tabular form in Fig. 5 has been then transformed into the concept taxonomy pictured in Fig. 9. This transformation also included adding the relevant property features to the concept features. Based on the groupings of the relevant features within the concepts of the taxonomy, ontological contexts for the most significant concept features (coloured green in Fig. 9) have been developed. This conceptualization work has been performed by:

- Developing a fragment of the descriptive theory of time around the concept feature and property features grouped with this concept
- Developing a conceptual model of this fragment as a UML class diagram
- Transforming the conceptual model to the formalized requirement specification in OWL (and SWRL in case a rule is needed to represent a feature)
- Documenting the requirement in a Wiki page

The results of these activities are illustrated below using the example of a TimeInstant concept feature.

The fragment of the descriptive domain theory elaborating the context of a time instant is presented below. It contains the basic definition of the concept feature, its place in the subsumption hierarchy, and also the description of the properties associated with this concept.

A time instant is a temporal element representing a point in time which has no duration. Having no duration is a qualifying feature of a time instant in difference to a time interval which has duration. From the other hand, a time instant is qualified by its position on the time line (temporal location), measured using the time stamp. There is only one time line on which a time instant is positioned. The time instants having equal timestamps but positioned on different time lines are regarded as different; before and after relations also do not work in this case.

An Origo is a specific kind of a time instant which, if exists for a particular time line, is placed at the beginning of times for this time line. Accodringly, there does not exist any other time instant on this time line which is positioned before the Origo. The segment of the Past on this time line is bounded at its beginning by the Origo.

A Present is a specific kind of a time instant which stands for now or current moment. By so saying, a Present is the boundary between the segments of the Past and Future, but not belonging neither to the Past, nor to the Future.

The basic property of a time instant is its (absolute) position on the time line with which the time instant is associated. This position is measured by the value (or the structured collection/bag of values) being the time stamp of the time instant. The format for the value / structured bag of values of a time stamp is specified by the time value domain.

A time stamp of a time instant may not be known. In this case, the (absolute) position of this time instant on a time line could be pointed to, with uncertainty, using qualitative temporal relations. For example, if it is known that an event occurred after 01/12/2017 and before 01/01/18 then the two time instants, $t_a$ and $t_b$, with these timestamps, can be added to a knowledge base. A time instant $t_e$ pointing to the temporal location of the event can then be asserted without time stamp, but having properties $after(t_e, t_a)$ and $before(t_e, t_b)$.

If a time stamp has a structure, this structure is commensurate to the available time units. The time units are chosen regarding to the precision (granularity) of the scale generated by the clock.

Relationships between the time instants on the same time line are specified to reflect their relativist positioning. Let $t_1$ and $t_2$ be any two time instants positioned on the same time line (T). Then, one and only one of the following three statements holds true reflecting the total linear ordering on the set of time instants: $before(t_1, t_2)$, $equals(t_1, t_2)$,

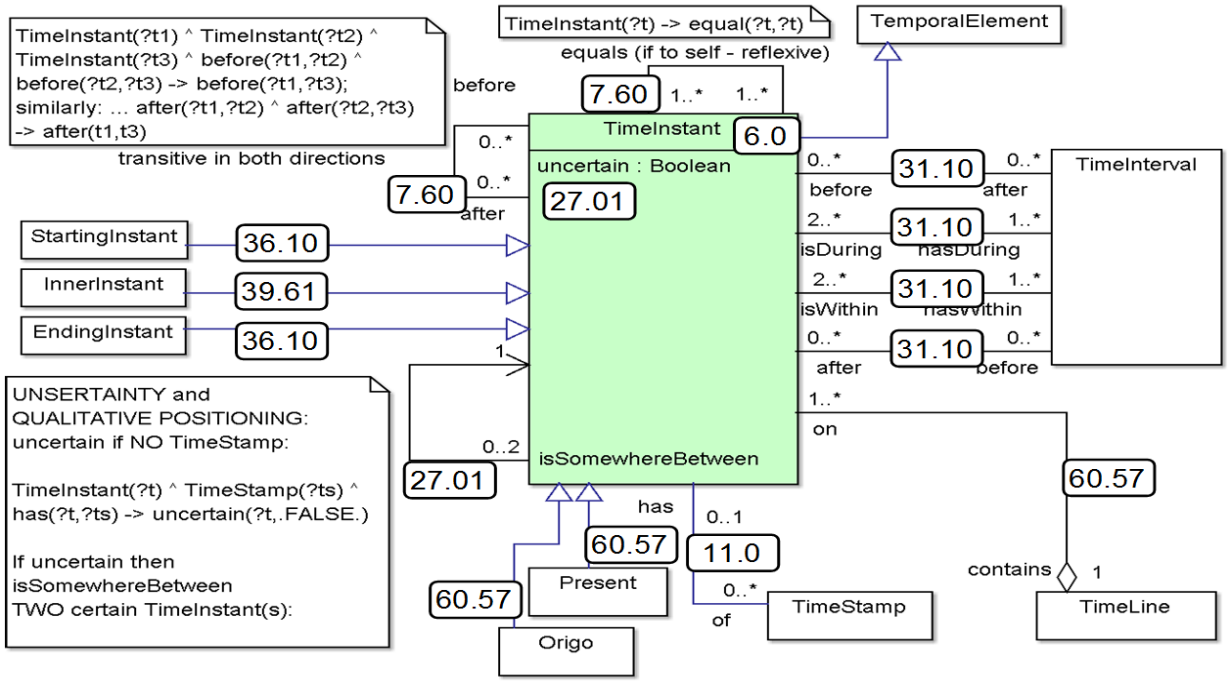



*Figure 10: The UML model of a TimeInstant. Feature significance scores are shown as the numbers in rounded rectangles. SWRL rules associated with the properties as restrictions are shown as comments. The overall significance score of the TimeInstant requirement is 509.96.*

$after(t_1, t_2)$. The total linear ordering imposes that the following hold true:

- Anisotropy:
  $\forall t_1, t_2 \in \mathrm{T}\ \ before(t_1, t_2) \leftrightarrow\ after(t_2, t_1)$
- Anisotropy: $\forall t_1 \in \mathrm{T}\ \ equals(t_1, t_1)$
- Transitivity:
  $\forall t_1, t_2, t_3 \in \mathrm{T}\ before(t_1, t_2) \wedge\ before(t_2, t_3) \rightarrow before(t_1, t_3)$
  $\forall t_1, t_2, t_3 \in \mathrm{T}\ \ before(t_1, t_2) \wedge\ \ after(t_2, t_3) \rightarrow after(t_1, t_3)$

It is considered that there could be no uncertainty in the relations between time instants. However, there could be uncertainty in the comparison of the timestamps of different time instants.

Relationships between the time instants positioned on different time lines can not be specified (directly) as there is no ordering established between the elements of different time lines.

Therefore, currently within this framework temporal relations can be inferred for time instants / intervals on the same time line. A rule / set of rules accounting for the above mentioned complications has to be set to infer if a time point on one timeline is before, after, or equal to a time point on the other time line in future work.

One possible way to reason about the relativist relationship between the time instants positioned on different time lines is to compare the values of their timestamps. This comparison is complicated by at least:

- The differences in the presence and relative positioning (offset) of the Origo points on these time lines
- The difference in the velocity of the time flow for different time lines
- The difference in the time units chosen for structuring the timestamps

The UML model of a time instant and the significance scores of its features are pictured in Fig. 10.

### 4.5 Evaluating the Fitness of OWL-Time to the Elicited Requirements

As suggested in Section 3.4, the mappings between the required features grouped in the formalized requirements and the corresponding elements of OWL-Time were specified for every significant requirement (Fig. 9): a TimeInstant and a TimeInterval. It was not possible to specify the mappings for a TimeLine and a Clock as these requirements were not implemented in OWL-Time. The mappings of TimeInstant are pictured in Fig. 11. In the figure, the mappings are given in a condensed form, without mentioning the names of f and e, as sources and tagets are easily identified by the arrows. Confidence factors are also not provided and are all equal to 1. Furthermore, as we are not interested in computing the gravitation forces in this paper, the spheres of influence of the features *(n-scores)* were neglected. The scores are given in round brackets as pairs of positive and negative votes.

The results of collecting votes and computing the fitness of OWL-Time to the elaborated TIME requirements are summarized in Table 4.

It may be stated that partially implemented features in an ontology raise more concern than the features that were not implemented at all. Indeed, if a feature has been implemented in part then the knowledge engineer has been made aware about the necessity of this feature by a requirement. Therefore, the missing bits of the feature semantics need to be added to the ontology. The features that are fully missing have not been required from the ontology before. Thus, it would be good to focus on the analysis of these features and the feasibility of their implementation.

As a straightforward recommendation, it may be mentioned that implementing the following five features with top significance scores will help to improve the fitness of OWL-Time by 233.41 points:

- Fuzziness – 65.92 + FuzzyInterval – 39.46
- TemporalStructure – 57.29
- TemporalFeature (high-level) – 27.01, including Uncertainty, (Un)Boundedness, Openness/ Closeness, Density (Discrete, Dense, Continuous)
- NonConvexInterval – 22.0
- PeriodicTemporalElement – 21.73

This will increase the fitness of OWL-Time to TIME requirements to 0.5768 by 17.97 percentage points.

It needs to be mentioned to conclude the discussion of this use case that the presented results have limited validity. Indeed, the fitness of OWL-Time was measured against the requirements reflecting the sentiment about time representation and reasoning by the TIME community. The results may have been different if another community and their representative document collection has been chosen.

## 5 Concluding Remarks

This paper showcased how the synergy of text mining, conceptual modelling, and ontology engineering techniques collected in one methodology helps transform the craft of ontology refinement to engineering. It reported on the use of the OntoElect methodology for evaluating the fitness of an existing ontology to the requirements of the knowledge stakeholders in the domain of Time Representation and Reasoning. It demonstrated, that a thorough routine for indirect elicitation, ensuring completeness, correctness of interpretation, using in ontology evaluation of these requirements is a must for Ontology Engineering.

In its motivating Section 2, the paper argued that both conceptual modelling and ontology engineering were in fact crafts to a substantial extent. The disciplines always claimed their careful attitude to the requirements of domain knowledge stakeholders. They did not however provide an objective and rigorous way to measure if: (a) all

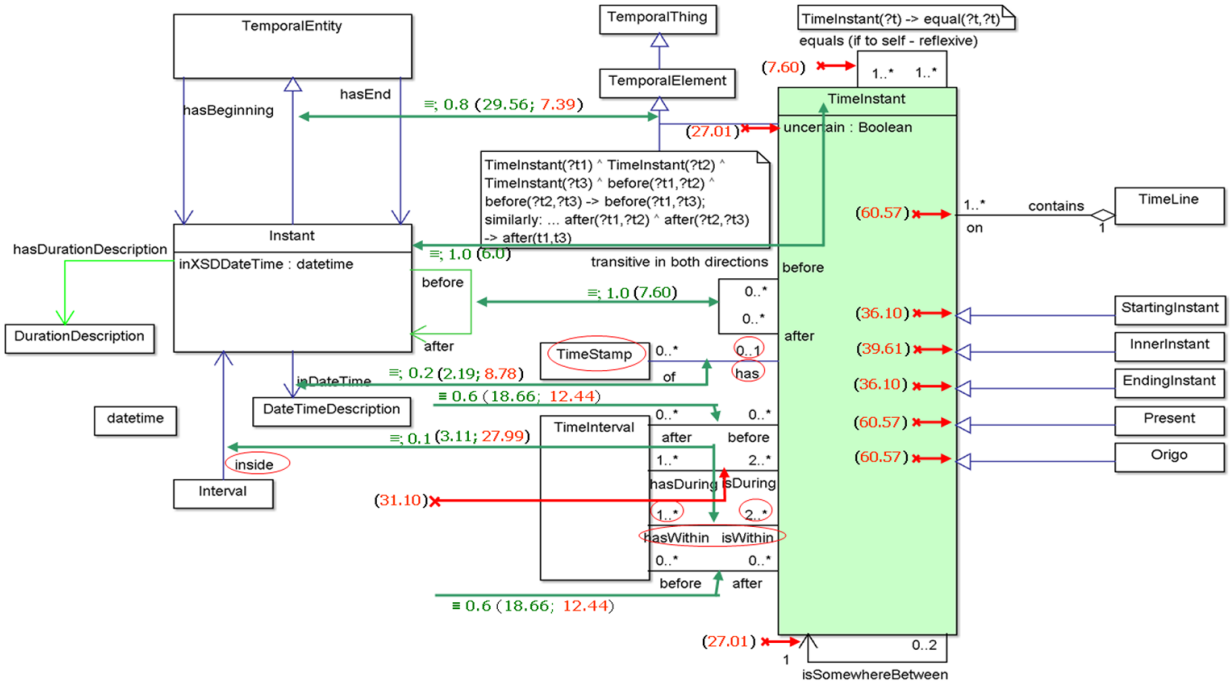



*Figure 11: The mappings of the TimeTnstant requirement to OWL-Time*

*Table 4: Fitness of OWL-Time to the 4 most significant TIME Requirements*

| Key Element | Fully Implemented Features | Partially Implemented Features | Missing Features |
|---|---|---|---|
| | Cumulative Significance Count | | |
| TimeInterval context **(1231.04)** | **351.59** | **97.05** / **103.15** | **679.25** |
| TimeInstant context **(509.96)** | **13.60** | **53.52** / **56.60** | **386.24** |
| TimeLine (TemporalStructure) **(57.29)** | — | — | **57.29** |
| Clock (TemporalDistanceMeasure + Clock) **(16.25)** | — | — | **16.25** |
| Total: | **515.76** | | **1298.78** |
| Fitness: | | **0.3971** | |

significant requirements were put on the table; (b) all these requirements were correct or correctly interpreted; and (c) the final product met these significant requirements satisfactorily. After looking at the related work facing these important research questions, the paper arrived at understanding that a synergy of different techniques and approaches from Automated Term Extraction, Conceptual Pre-design, Ontology Evaluation is required to enable a coherent processing pipeline with enough maturity and rigour to answer these questions.

The paper reported on the research which has been inspired by the State–of–the–Art contributions in the above mentioned areas. The ideas and techniques that facilitated the development of OntoElect methodology in its current shape were:

- For an indirect approach to feature elicitation and answering the completeness question, a method to measure the terminological completeness of the document collection by analysing the *saturation* of terminological footprints of the incremental slices of the collection proposed by Tatarintseva et al. (2013).
- For ensuring the correct interpretation of the features as requirements, at least two ideas of KCPM by Kop et al. (2004) were valuable for reuse: (i) requirements are formalized to become closer to conceptual models; and (ii) requirements are focused on the terminology elaborated within the expert community in a domain
- For developing a metric of how well an ontology meets the requirements, the proposal by Gangemi et al. (2006) to measure ontology fitness for usability profiling appeared to be quite inspiring as it has been used to develop a gravitation-based technique to measure fitness against formalized requirements through the use of mappings.

The paper in its overview of OntoElect in Section 3 reported on how these motivating ideas have been further developed and refined in each of the processing phases of the methodology: Feature Elicitation, Requirements Conceptualization, and Ontology Evaluation.

With respect to the **feature elicitation** pipeline, the paper explained the technique used for discovering a saturated sub-collection of documents. Here, terminological difference *(thd, thdr, eps)* was used as a metric to detect saturation. It argued about the way to discover the documents with the highest terminological impact – the decisive minority sub-collection – using the citation frequency as the basic metric. It explained how to sublimate the extracted terms to the set of required features using the information about term significance in the form of numeric scores.

Regarding **requirements conceptualization and formalization**, the paper offered the sequence of activities to create formalized ontological fragments for requirements. It proposed a way to categorize and group the required features. Based on these groupings, it explained how to build the feature taxonomy using subsumptions, part-whole relationships, and memberships among the required features. Furthermore, it explained why and how significance scores of the required features ought to be refined by accounting for their propagation through inheritance. As a next step, the development of the feature taxonomy was suggested which was helpful for prioritizing the features based on their refined significance scores. The guidelines have further been given on how to group and aggregate the required features in the proper ontological fragment and in a harmonized way.

For **ontology evaluation**, the paper proposed to use the allusion of a gravitation grid and field for measuring the difference of the ontology to the requirements. Equivalence mappings, incorporating feature significance scores, were regarded as atomic gravitation forces denoted as positive and negative community votes with respect to the ontology. It has been proposed to measure the overall fitness of the ontology to the requirements as the ratio of positive to negative votes.

In Section 4, the methodology has been evaluated by applying it to measuring the fitness of the W3C OWL- Time ontology to the requirements

elicited from the representative collection of research papers of the TIME symposia series. The three phases of the methodology were applied to this collection. As a result, it has been shown – in numbers – that OWL-Time meets the TIME community requirements only marginally. The paper suggested 5 major refinements to the ontology which may substantially increase its fitness. Interestingly, these proposed refinements differ noticeably from those elaborated by the experts in the W3C SDW WG by brainstorming.

Finally, for indicating the plans for the future work on OntoElect, it may be noted that the instrumental support for conceptualization and evaluation phases of the methodology is under development. Having this instrumental support will allow to lower the effort of a knowledge engineer in checking if (s)he really met the requirements of the knowledge stakeholders in the domain of the ontology under refinement.

At the time of writing, the W3C Recommendation of the Time ontology in OWL has appeared (Cox et al. 2017). Therefore, one more plan for the future work is to evaluate this updated revision against the TIME requirements.

## 6 Acknowledgements

The author would like to express his gratitude to Prof. Heinrich C. Mayr and the members of his Conceptual Predesign (KCPM) taskforce for the inspiration on the high-level idea of OntoElect – using stakeholder contributions to elicit requirements indirectly; and also for their approach to ensure that the collected requirements are interpreted correctly.

The research leading to this paper has been done in part in in frame of FP7 Marie Curie IRSES SemData project (http://www.semdata-project.eu/), grant agreement No PIRSES-GA-2013-612551.

The author would also like to acknowledge generous help of several colleagues who made achieving the reported results possible. Olga Tatarintseva contributed to the development of the OntoElect methodology at an early stage. Victoria Kosa helped quite significantly with extracting the required features. Dr. Andreas Harth helped a lot with choosing and substantiating the W3C SDW WG use case as the member of this expert group. Dr. Natalya Keberle helped in transforming the conceptual (UML) models to OWL 2 DL. Dr. Sotiris Batsakis and Prof. Frederic Mallet contributed their time and expertise in the discussions of the fragments of the descriptive theories of timelines, time intervals, time instants, and clocks. The author would also like to thank the anonymous reviewer who suggested to change ontologies to conceptual models to pass the test of time.

## References

Booch G., Jacobson I., Rumbaugh J. (2000) OMG Unified Modeling Language specification

Burton-Jones A., Storey V. C., Sugumaran V., Ahluwalia P. (2005) A semiotic metrics suite for assessing the quality of ontologies. In: Data and Knowledge Engineering 535(1) October 2005, pp. 84–102

Chien L.-F., Chen C.-L. (2001) Incremental extraction of domain-specific terms from online text resources. In: Bourigault D., Jacquemin C., L'Homme M.-C. (eds.) Recent advances in computational terminology. Natural Language Processing Vol. 2. John Benjamins Publishing, Amsterdam/Philadelpia, pp. 91–109

Cox S., Little C., Hobbs J. R., Pan F. (2006) Time Ontology in OWL. W3C working draft.. Ontology Engineering Patterns Task Force. Semantic Web Best Practices and Deployment Working Group. World Wide Web Consortium (W3C). http://www.w3.org/TR/2006/WD-owl-time-20060927/

Cox S., Little C., Hobbs J. R., Pan F. (2017) Time Ontology in OWL. W3C Recommendation. OGC 16-071r2. Ontology Engineering Patterns Task Force. Semantic Web Best Practices and Deployment Working Group. World Wide Web Consortium (W3C). https://www.w3.org/TR/2017/REC-owl-time-20171019/

Ermolayev V. (2015) The law of gravitation for ontologies and domains of discourse. In: Computer Science Journal of Moldova 23(2), pp. 209–236

Ermolayev V., Batsakis S., Keberle N., Tatarintseva O., Antoniou G. (2014) Ontologies of time: review and trends. In: International Journal of Computer Science and Applications 11(3), pp. 57–115

Fliedl G., Kop C., Mayr H. C. (2005) From textual scenarios to a conceptual schema. In: Data and Knowledge Engineering 535(1), pp. 20–37

Gangemi A., Catenacci C., Ciaramita M., Lehmann J. (2006) Modelling ontology evaluation and validation. In: Sure Y., Domingue J. (eds.) Proc. ESWC 2006. LNCS Vol. 4011. Springer-Verlag, Berlin/Heidelberg, pp. 140–154

Gómez-Pérez A., Fernández-López M., Corcho O. (2004) Ontological engineering. Springer-Verlag, London

Kop C., Mayr H. C., Zavinska T. (2004) Using KCPM for defining and integrating domain ontologies. In: et al C. B. (ed.) Proc. WISE Workshops. LNCS Vol. 3307. Springer-Verlag, Berlin/Heidelberg, pp. 190–200

Kosa V., Chaves-Fraga D., Naumenko D., Yuschenko E., Badenes-Olmedo C., Ermolayev V., Birukou A. (2017) Cross-evaluation of automated term extraction tools. TS-RTDC-TR-2017-1. Dept of Computer Science, Zaporizhzhia National University, Ukraine

Motik B., Patel-Schneider P. F., Parisa B. (2012) OWL 2 Web Ontology Language. Structural specification and functional-style syntax (Second Edition).. http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/

Newton I. (1999) The principia, mathematical principles of Natural Philosophy, a new translation. By I Bernard Cohen and Anne Whitman, preceded by "A guide to Newton's principia" by I Bernard Cohen. University of California Press

Pinto H. S., Tempich C., Staab S., Sure Y. (2004) DILIGENT: towards a fine-grained methodology for distributed, loosely-controlled and evolving engineering of ontologies. In: de Mántaras R., Saitta L. (eds.) 16th European Conf. on Artificial Intelligence (ECAI). IOS Press

Poveda-Villalón M. (2016) Ontology evaluation: a pitfall-based approach to ontology diagnosis. PhD thesis, Universidad Politécnica de Madrid

Presutti V., Blomqvist E., Daga E., Gangemi A. (2012) Pattern-based ontology design In: Ontology engineering in a networked world. Suárez-Figueroa M. C., Gómez-Pérez A., Motta E., Gangemi A. (eds.) Springer-Verlag, Berlin/Heidelberg, pp. 35–64

Schreiber G. (1999) Knowledge engineering and management: the CommonKADS methodology. the MIT Press, Cambridge, Massachusetts

Suárez-Figueroa M. C., Gómez-Pérez A., Motta E., Gangemi A. (2012) Ontology engineering in a networked world. Springer-Verlag, Berlin/Heidelberg

Sure Y., Staab S., Studer R. (2004) On-To-Knowledge methodology (OTKM) In: Handbook on ontologies Staab S., Studer R. (eds.) Series of Handbooks in Information Systems Springer-Verlag, Berlin/Heidelberg, pp. 117–132

Tatarintseva O., Ermolayev V., Keller B., Matzke W.-E. (2013) Quantifying ontology fitness in OntoElect using saturation- and vote-based metrics. In: et al. V. E. (ed.) Revised selected papers of ICTERI 2013, CCIS. CCIS Vol. 412. Springer-Verlag, Berlin/Heidelberg, pp. 136–162

Vrandecic D. (2010) Ontology evaluation. PhD thesis, Fakultät für Wirtschaftswissenschaften, KIT, Karlsruhe, Germany

Wong W., Liu W., Bennamoun M. (2012) Ontology learning from text: a look back and into the future. In: ACM Comput. Surv. 44(4) Article 20, 36 pages

# A Universal Ontology-based Approach to Data Integration

Antoni Olivé*,a

[a] Universitat Politècnica de Catalunya – BarcelonaTech. Catalonia

Abstract. *One of the main problems in building data integration systems is that of semantic integration. It has been acknowledged that the problem would not exist if all systems were developed using the same global schema, but so far, such global schema has been considered unfeasible in practice. However, in our previous work, we have argued that given the current state-of-the-art, a global schema may be feasible now, and we have put forward a vision of a Universal Ontology (UO) that may be desirable, feasible, and viable. One of the reasons why the UO may be desirable is that it might solve the semantic integration problem. The objective of this paper is to show that indeed the UO could solve, or at least greatly alleviate, the semantic integration problem. We do so by presenting an approach to semantic integration based on the UO that requires much less effort than other approaches.*

Keywords. Universal Ontology • Data Integration • Mediated Schema • Conceptual Modelling.

## 1 Introduction

The goal of data integration systems is to offer uniform access to a set of autonomous and heterogeneous data sources. Building such systems is difficult for system, logical and social reasons (Doan et al. 2012, p. 6). One of the main problems is that of semantic integration, which arises from the fact that the conceptual schemas of the data sources to be integrated have been developed independently. For some authors, despite its pervasiveness and importance, semantic integration remains an open and extremely difficult problem (Batini et al. 1992, Park and Ram 2004).

Several authors have acknowledged in the past that the semantic integration problem would not exist, or at least it would be greatly alleviated, if all systems were developed using the same global schema (Madnick 1996, Uschold 2000, Mena et al. 2000, Grüninger and Uschold 2004). However, such global schema has been considered unfeasible in practice, and therefore, as far as we know, data integration in the context of a global schema has not been explored in the same level of detail as in other contexts.

In a recent paper (Olivé 2017), we have argued that a global schema may have been unfeasible in the past, but it is no longer the case now. We have put forward a vision of a global schema, called the Universal Ontology (UO), that is desirable, feasible in the current state of the art, and viable.

One of the reasons why the UO is desirable is that it might solve the semantic integration problem in data integration systems. However, as we have said, little is known about the use of the UO in those systems and its effectiveness in solving that problem. We review some of the work that has been done in the section on related work.

The objective of this paper is to show that indeed the UO solves, or at least greatly alleviates, the semantic integration problem. We do so by presenting an approach to data integration based on the UO that requires much less effort than other approaches. The approach requires that the developers of the data sources define the semantic mappings between the data source schemas and

* Corresponding author.
E-mail. antoni.olive@upc.edu
This work has been partially supported by the Ministerio de Economía y Competitividad, under project TIN2014-52938-C2-2-R.

the UO. From these semantic mappings, the other semantic components of a data integration system (mediated schema, source descriptions) can in principle be automatically generated.

The rest of the paper is structured as follows. Next section briefly describes the scope, the kinds of the concepts, the elements that comprise the specification of each concept of the UO, and the concept composition operators. Section 3 explains how to define the mappings between data sources and the UO. Section 4 presents the framework of the data integration system, and the semantic components to be generated. Section 5 presents a method for the generation of the mediated schema and the source descriptions. Section 6 discusses some related work. Finally, section 7 summarizes the conclusions. Throughout the paper we use the example introduced in (Doan et al. 2012, p. 67) with minor modifications.

This paper has been written as a sincere tribute to Heinrich C. Mayr on occasion of his retirement. During his professional life as researcher, he has made substantial contributions to the conceptual modelling field since the beginning (Lockemann et al. 1979) until recently (Michael and Mayr 2017) and, at the same time, he has had an active involvement in many organizational activities that have helped to build a strong and friendly conceptual modelling community.

## 2 The Universal Ontology

In this section, we summarize the characteristics of the UO proposed in Olivé (2017) that we will need in this paper. We explain first the specification of the UO concepts and then the concept composition operators.

### 2.1 Concept specification

The objective of the UO is to allow the publication, search, and reading by people and machines of any fact of any domain, using an integrated set of all existing concepts. The UO comprises three kinds of concepts: entity types, datatypes and binary properties. There are two kind of properties: object properties, which link entities to entities, and datatype properties, which link entities to data values. Properties have a direction, from subject (domain) to object (range).

We have argued that WordNet (Miller 1995), among others, could be the basis of a substantial part of the UO. For this reason, the examples of this paper are taken from WordNet, although we make an ad hoc use of datatypes.

The specification of each concept includes at least:

- The kind of the concept.
- The concept identifier. Each concept should have a natural language-independent, unique, and immutable identifier. In the examples of this paper, we will use as concept identifier the word# sense number of the concept in WordNet, such as the noun *Movie#1*.
- The name and synonym(s) of the concept in each natural language spoken by the UO users. The names of the concepts need not be unique. In general, the name of an entity or data type must be a noun phrase, while the name of a property must be a verb phrase or a noun phrase. When the name of a property is a noun phrase, the property is seen as an attribute (characteristic, feature . . . ) of the subject. Most of the properties in the examples of this paper are attributes. We will assume that their identifier has the general form *HasType*, where *Type* is the identifier of an entity or datatype. For example, *HasTitle#2*.
- The definition of the concept. It may be a natural language description or a derivation rule in some formal language.
- The supertypes of the concept (*IsA* relationships).
- The analytical constraints that the instances of the concept must satisfy to be considered universally valid. Among these constraints there are the allowed domain and range of properties, and the disjointness constraints of concepts.
- The (meta-)entity types of which an entity type is an instance (*InstanceOf* relationships).

### 2.2 Concept composition

The UO described above specifies only a limited (even if very large) number of concepts. However, it is a fact that using an appropriate set of composition operators we could compose a limitless number of concepts from them. We call core UO the explicitly defined ontology, and extended UO the set of concepts that could be composed from the core. The full UO would then be the union of the core and extended parts.

In the examples of this paper, we will use only the reification operator described in the following. Other operators have been defined in Olivé (2017).

The reification operator is well known in conceptual modelling. Let $P$ be a property, $E_1$ an entity type that is in the domain of $P$ and $E_2$ an entity or data type that is in the range of $P$. Then, $E$ = *Reif*($E_1$,$P$,$E_2$) is an entity type that corresponds to the reification of $P$ with domain $E_1$ and range $E_2$. An instance of $E$ corresponds to an instance of $P$ such that its subject is an instance of $E_1$ and its object is an instance of $E_2$.

The UO includes two properties that can be used when needed:

- *HasSubject*, which links a reification $E$ to its subject $E_1$.
- *HasObject,* which links a reification $E$ to its object $E_2$.

## 3 Mapping relational schemas to the UO

Our approach to data integration requires the mapping of the data sources to the UO. This mapping is done for each data source, independently of other data sources with which it might be integrated. In fact, it might be justified to do this mapping as a means for documenting a data source schema.

In this paper, we will focus only on data sources that are relational databases.

### 3.1 Anchors

Each relational schema $R$ must be mapped to an entity type $E$ of the full UO, which we call the anchor of $R$, written as *anchor*($R$) = $E$ (An et al. 2006). The meaning is that a tuple of $R$ represents data about an instance of $E$. Each tuple of $R$ corresponds to a different instance of $E$. In a given database, there may be several relational schemas whose anchor is the same entity type.

In the example, there are six relational databases, $S_1$ to $S_6$. The relational schemas of these databases and their corresponding anchors are the following:

$S_1$:
  $R_1$: *Movie*(*MID*,*title*) *anchor*($R_1$) = *Movie#1*
  $R_2$: *Actor*(*AID*,*firstName*,*lastName*,*nationality*, *yearOfBirth*) *anchor*($R_2$) = *Actor#1*
  $R_3$: *ActorPlays*(*AID,MID*)
  *anchor*($R_3$) =
  *Reif*(*Movie#1,HasActor#1,Actor#1*)
  $R_4$: *MovieDetails*(*MID*,*director*,*genre*,*year*)
  *anchor*($R_4$) = *Movie#1*
$S_2$:
  $R_5$: *Cinemas*(*place,movie,start*)
  *anchor*($R_5$) = *Show#3*
$S_3$:
  $R_6$: *Reviews*(*title,date,grade,review*)
  *anchor*($R_6$) = *Review#2*
$S_4$:
  $R_7$: *MovieGenres*(*title,genre*)
  *anchor*($R_7$) = *Movie#1*
$S_5$:
  $R_8$: *MovieDirectors*(*title,dir*)
  *anchor*($R_8$) = *Movie#1*
$S_6$:
  $R_9$: *MovieYears*(*title,year*)
  *anchor*($R_9$) = *Movie#1*

Note that, in the example, the anchor of most relational schemas is *Movie#1*. The anchor of $R_2$ is *Actor#1*. The anchor of $R_3$ is an entity type of the extended UO. In this case, it is the result of the reification operator, introduced in the previous section. Each tuple of $R_3$ corresponds to a participation of an actor in a movie. The anchor of $R_5$ is *Show#3* ("a social event involving a public performance or entertainment"). The anchor of $R_6$ is *Review#2* ("an essay or article that gives a critical evaluation (as of a book or play)").

### 3.2 Mapping attributes

In a relational schema $R$, each attribute $A$ of type $T$ maps to a datatype property $P$ of the full UO. The domain of $P$ for $A$ in $R$ is the same of $P$ in the UO or a subtype of it. The range of $P$ for $A$ in $R$ is $T$, which must be the same of $P$ or a subtype of it.

We write $map(A) = P(D,Rg)$ to indicate that attribute $A$ maps to property $P$ and that the domain of $P$ in $R$ for $A$ is $D$, and its range $Rg$. For example, the mapping of attribute *title* of $R_1$ of type *String* is:

$map(title) = HasTitle\#2(Movie\#1,String)$

This means that attribute *title* of $R_1$represents the value of property *HasTitle#2* of the instances of *Movie#1*, which is the anchor of $R_1$. The mapping would be the same for attribute *title* in $R_7$, $R_8$ and $R_9$.

For mapping purposes, we must distinguish two kinds of attributes in $R$, which we call direct and indirect. A direct attribute $A$ is an attribute of the anchor of $R$, and, therefore, it maps to a data type property $P$ whose domain is the anchor of $R$ and its range is $T$.

An example is the attribute *title* as indicated above. Another example may be the attribute *start* of $R_5$ whose type is assumed to be *Time*. It maps to the datatype property *HasShowTime#1*:

$map(start) = HasShowTime\#1(Show\#3,Time)$

An indirect attribute $A$ of type $T$ of $R$ is an attribute of an entity type $O$ related to the anchor of $R$ by means of an object property $P_1$. $O$ is called an indirect entity type of $R$. In this case, we write:

$map(A) = (P_1(anchor(R),O),P(O,T))$

For example, in $R_4$, attribute *director* is the name of the director of a movie, that is:

$map(director) =$
$(HasDirector\#4(Movie\#1,Director\#4),$
$HasName\#1(Director\#4,String))$

In this case, an indirect entity type of $R_4$ is *Director#4*. Note that *HasName#1* refers to the name of a director, not that of the anchor of $R_4$ (movie). The same mapping applies to attribute *dir* of $R_8$.

As another example, we have attribute *year* of $R_4$ and $R_9$, which refers to the year in which a movie was released. Both map to:

$map(year) =$
$(HasRelease\#2(Movie\#1,Release\#2),$
$HasDate\#1(Release\#2,Year))$

In $R_5$ we find two indirect attributes:

$map(place) =$
$(HasCinema\#2(Show\#3,Cinema\#2),$
$HasName\#1(Cinema\#2,String))$
$map(movie) =$
$(Show\#1(Show\#3, Movie\#1),$
$HasTitle\#2(Movie\#1,String))$

When the anchor of $R$ is a reification, then $R$ has one or more attributes that map to the subject, and one or more attributes that map to the object of the reification.

In the example, for $R_3$ we have

$map(MID) =$
$(HasSubject(E,Movie\#1),$
$HasIdentifier\#1(Movie\#1,Integer))$
$map(AID) =$
$(HasObject(E,Actor\#1),$
$HasIdentifier\#1(Actor\#1,Integer))$

where $E = Reif(Movie\#1,actor\#1,Actor\#1)$. Note that in this case the two attributes are indirect.

### 3.3 Identifiers

Each relational schema must have at least one identifier of its anchor, and may have one identifier of each indirect entity type, if any. An identifier consists of one or more attributes whose values

identify the corresponding instances. An identifier is simple if it consists of only one attribute, and composite if otherwise.

For example, the anchor of $R_1$ has two identifiers, both simple. The first is the attribute *MID*, and the second is the attribute *title*. In $R_2$, the anchor has two identifiers, one simple and the other composite. The simple is attribute *AID*. The composite consists of attributes *firstName* and *lastName*. In $R_4$, the indirect object *Director#4* has a simple identifier (attribute *director*).

In data integration, it is important to distinguish between local and global identifier attributes, depending on whether they are locally or globally known. The value of a local identifier attribute identifies an entity in a way that is known only in the context of a data source, while that of a global one identifies an entity in a way that is or may be known globally.

For example, in $R_1$, *MID* is local, while *title* is global. The values of *MID* identify movies in a way that is known only in $S_1$, while the values of *title* identify movies in all data sources.

As another example, the anchor of $R_6$ has a composite identifier, consisting of two global identifier attributes *title* and *review*, whose mappings are:

*map*(*title*) =
(*Review#2*(*Review#2*,*Movie#1*),
*HasTitle#2*(*Movie#1*,*String*))
*map*(*review*) = *HasText#1*(*Review#2*,*String*)

Note that in the above example, *title* is an indirect attribute whose indirect entity type is *Movie#1*. The indirect object property is written as *Review#2*, which in this case is the WordNet verb synset *review#2* (" appraise critically").

When the anchor of a relational schema $R$ is a reification, then *anchor*($R$) has normally a composite identifier consisting of two attributes: one for the subject and one for the object of the reification. In the example of $R_3$, *anchor*($R_3$) has a composite identifier, consisting of the two local identifier attributes *MID* and *AID*.

### 3.4 Assessment

In general, designers and users of relational databases know the anchor and the properties of their relational schemas. As we have seen, our approach requires that anchors and properties are related to the corresponding concepts in the UO. This should be easy if the concepts are in the core part, and may be not so easy if they must be composed. Anyway, this must be done only once per relational schema and, on the other hand, it is useful as a documentation of the semantics of the schema. We assume that the UO will be large enough to include most of the possible anchors and properties. When this is not the case, the corresponding relational schema will need to be manually integrated.

## 4 Data integration framework

In this section, we briefly introduce the data integration framework in which we place our work. We take as a basis the work reported in Lenzerini (2002) and Doan et al. (2012).

The goal of a data integration system is to combine the data residing at different sources, and to provide the users with a unified view of these data. The unified view is represented by the mediated schema, which is a reconciled view of all data that can be queried by the user. The main components of a data integration system are the mediated schema, the data sources, and the mappings of the data sources to the mediated schema, also called source descriptions.

There are three main approaches for specifying the mappings: Local-as-View (LAV), Global-as-View (GAV) and Global-and-Local-as-View (GLAV). In our approach we use LAV mappings. Such mappings associate each element of a data source to the mediated schema, independently of any other data sources.

The main tasks in the design of a data integration system are to design the mediated schema and to define the mappings between the sources and the mediated schema. In the next section, we describe how these tasks can be performed when the data sources are relational databases whose

schemas have been mapped to the UO as indicated in the previous section.

## 5 The mediated schema and source descriptions

When the data source schemas are mapped to the UO as indicated in section 3, the mediated schema and the source descriptions can be obtained as indicated in the following. We explain first the mediated schema and then the source descriptions.

### 5.1 Mediated schema

In our approach, we assume that the mediated schema must include all entity types, datatypes and properties represented in the local data sources.

In the example (Doan et al. 2012, p. 67), the mediated schema is defined by the following four relation schemas:

$M_1$: *Movie*(*title*,*director*,*year*,*genre*)
$M_2$: *Actors*(*title*,*name*)
$M_3$: *Plays*(*movie*,*location*, *startTime*)
$M_4$: *Reviews*(*title*,*rating*,*description*)

Note that the $R_2$ attributes *nationality* and *yearOfBirth* are not included in the mediated schema. The same happens with the local identifier attributes *MID* and *AID* in $S_1$, and the attribute *date* of $R_6$.

For simplicity's sake, and without loss of generality, we define the mediated schema using the logic formalism (An et al. 2006, p. 6); (Olivé 2007, ch. 2,3). Entity types and data types are represented by means of unary predicates, while properties are defined by means of binary predicates. *IsA* relationships, constraints and queries will be represented by first-order logic expressions.

The mediated schema $M$ consists of a set of entity and data types, with their *IsA* relationships and disjointness constraints, and a set of properties, with their domain and range constraints. We deal first with the entity types and then with the properties. Data types are treated similarly to entity types.

**Entity types**. The entity types of $M$ are the set of anchors and indirect entity types of all relational schemas of all data sources, and their entailed types, if any.

The *IsA* relationships of $M$ are those of the entailed types and those defined in the UO. If $E_1$ and $E_2$ are two entity types in $M$ and there is in the UO a direct or indirect $E_1$ *IsA* $E_2$ then there must be also a direct or indirect $E_1$ *IsA* $E_2$ in $M$.

The disjointness constraints of $M$ are those defined in the UO. If $E_1$ and $E_2$ are two entity types in $M$ and there is in the UO a direct or indirect disjointness between $E_1$ and $E_2$ then there must be also a direct or indirect disjointness between $E_1$ and $E_2$ in $M$.

Using our approach, the entity types of the mediated schema corresponding to the anchors are: *Movie#1, Actor#1, Reif (Movie#1,HasActor#1,Actor#1), Show#3*, and *Review#2*, and those corresponding to the indirect entity types are: *Country#1, Director#4, Cinema#2* and *Release#2*. In this example, we will ignore the *IsA* relationships and the disjointness constraints.

**Properties**. The properties of $M$ are the set of mapped properties of all relational schemas of all data sources, and their entailed properties, if any. The *IsA* relationships and the disjointness constraints of the properties, which is this example we will ignore as before, are represented similarly to those of the entity types.

Different attributes of the relational schemas of the data sources may map to the same property $P$ of the UO, with the same or different pairs of domain $D$ and range $Rg$. A pair $D$, $Rg$ of $P$ is called a realization of $P$ (Olivé 2007, ch. 7).

Each realization is represented in the mediated schema by a distinct predicate. For simplicity, we will name $P[D,Rg]$ the binary predicate corresponding to property $P$ with domain $D$ and range $Rg$. For example:

*HasTitle#2* [*Movie#1, String*]
*HasFirstName#1* [*Actor#1, String*]
*HasLastName#1* [*Actor#1, String*]

There is one exception to the above rule, which concerns local identifier attributes like *MID* in $R_1$, $R_3$ and $R_4$, whose mapping is:

*map*(*MID*) = *HasIdentifier#1*(*Movie#1*,*Integer*)

This attribute cannot be represented in the mediated schema by the binary predicate *HasIdentifier#1* [*Movie#1, Integer*] because it can be used only in data source $S_1$. The solution we propose, inspired in (Fowler 1997, p. 88), UN/CEFACT (2009) and SAP (2016), consists in using in these cases a quaternary predicate, which may have the same name as before. The meaning of an atomic sentence such as

*HasIdentifier#1*[*Movie#1*,*Integer*](*m*,*i*,*a*,*s*)

is that movie *m* has identifier *i* according to the rules defined by the agency *a* in the identification scheme *s*.

## 5.2 Source descriptions

Once we have obtained the mediated schema, we can automatically generate the source descriptions. We will use LAV mappings.

The general form of a LAV mapping will be:

$R(X) \rightarrow \exists\, !y\, (E(y) \wedge \phi(X,y))$

where $R$ is a relational schema, $X$ the set of attributes of $R$, $E$ the anchor of $R$, and $\phi(X,y)$ a formula over the predicates of the mediated schema. The variables in $X$ are implicitly universally quantified in front of the formula. The mapping states that each tuple of $R$ with attributes $X$ maps to one and only one instance $y$ of its anchor $E$ for which $\phi(X,y)$ holds.

For example, the source description of $R_7$ is:

*MovieGenres*(*title,genre*) → ∃ !*y*(*Movie#1*(*y*) ∧
*HasTitle#2*[*Movie#1,String*](*y*,*title*) ∧
*HasGenre#1*[*Movie#1,String*](*y*,*genre*))

Each direct attribute $A$ of type $T$ for which *map*($A$) = $P$($D$,$Rg$) has the atom $P$[$D$,$Rg$]($y$,$t$) in $\phi$. This expresses that if $t$ is the value of attribute $A$ in a tuple then $P$[$D$,$Rg$]($y$,$t$) must be true. In the above example, we have:

*map*(*title*) = *HasTitle#2* (*Movie#1*,*String*)

and therefore

*HasTitle#2*[*Movie#1,String*](*y*,*title*) must be true.

When an attribute is a local identifier, such as *MID* in $R_1$, $R_3$ and $R_4$, the corresponding atom is a quaternary predicate as explained above. In this way, the source description of $R_1$ is:

*Movie*(*mid*,*title*) → ∃ !*y*(*Movie#1*(*y*) ∧
*HasIdentifier#1*[*Movie#1*,*Integer*]
(*y*,*mid*,'$S_1$','*Movie*') ∧
*HasTitle#2*[*Movie#1,String*](*y*,*title*))

where we have assumed that '$S_1$', the name of the data source, is the name of the agency and '*Movie*' the name of the identification scheme. Both are constants and, of course, they can be changed.

An indirect attribute $A$ such that

*map*($A$) = ($P_1$($E_1$,$O$),$P$($O$,$T$))

is represented in $\phi$ by the formula

∃ !$z$ ($O$($z$) ∧ $P_1$[$E_1$,$O$]($y$,$z$) ∧ $P$[$O$,$T$]($z$,$t$))

As an example, consider $R_8$ in which we have

*map*(*director*) =
(*HasDirector#4*(*Movie#1*,*Director#4*),
*HasName#1*(*Director#4*,*String*))

where *HasName#1* is a simple global identifier of *Director#4*. The source description of $R_8$ is then:

*MovieDirectors*(*title,dir*) →
∃ !*y*(*Movie#1*(*y*) ∧

*HasTitle#2*[*Movie#1,String*](*y*,*title*) ∧
∃ !*z*(*Director#4*(*z*) ∧
*HasDirector#4*[*Movie#1*,*Director#4*](*y*,*z*) ∧
*HasName#1*[*Director#4,String*](*z*,*dir*)))

When an indirect attribute maps to an entity type that has a composite identifier, then all attributes in $R$ of that identifier are represented together in $\phi$ after the initial part:

$$\exists\, !z\, (O(z) \wedge P_1[E_1,O](y,z) \wedge \ldots$$

### 5.3 Assessment

We have shown that the mediated schema and the source descriptions can be automatically generated from the mappings of the relational schemas to the UO. The main drawback we see is that some predicate names of the mediated schema may be not user-friendly, which is a problem if there are human users. In this case, the predicate names will need to be manually improved by the designers.

## 6 Related work

In this section we review two precursor systems that used a kind of universal ontology for data integration: Carnot and SIMS.

As far as we know, Carnot (Collet et al. 1991; Huhns et al. 1993) was the first system that maps local schemas to a pre-existent global schema, which in this case it is the Cyc ontology. Each local schema is mapped to Cyc, independently of other local schemas. Carnot includes a tool (Model Integration Software Tool) that assists users in finding the correspondences of each concept of the local schema with a Cyc concept. Using those correspondences, the tool automatically generates articulation axioms, which formally state the mapping between the instances of the local source and Cyc's knowledge base.

In Carnot there are not mediated schemas. The local schemas are integrated into Cyc, extending it if needed. The consequence is that the data content of integrated system is not explicit, and therefore users may find difficult to query it (Collet et al. 1991, p. 62).

The work most closely related to ours is SIMS (Arens et al. 1993; Arens et al. 1996). SIMS uses two kinds of models, both written in the Loom language. The first is a domain model, which describes the classes in the domain and their relationships. The second is the data source model, which describe the classes and relationships in each data source.

The mapping between a data source model and the domain model is done by defining a data source link, IS-link, between the concepts and roles in both models. The meaning of an IS-link between a data source class and a domain class is that the two classes contain exactly the same set of individuals, although the data source class might contain only a subset of the attributes for the class. The links between the roles indicate that those roles have the same meaning for the linked classes.

In SIMS, the minimal model of a data source is a model that includes a data source model and enough of a domain-level model to exactly cover the data source model. The mediated schema, called minimal model, is then the union of all minimal models for all the data sources available to the system. Informally, the minimal model is the smallest model that can describe the semantics of, and provide access to, the entire contents of the available data sources.

SIMS shows that, when a domain model is available and the data sources are mapped to it, the design of the mediated schema and the source descriptions is easier. Our approach is similar to that of SIMS, but we do not need to model the data sources and the LAV mappings of the data sources to the universal ontology can be more expressive.

## 7 Conclusions

We have tried to show that the universal ontology (UO) solves, or at least greatly alleviates, the semantic integration problem. To this end, we have presented an approach to data integration based on the UO.

Our approach requires that the developers of the data sources define the mappings between the data source schemas and the UO. We have argued

that this is easy if the concepts are in the core part of the UO, although it may be not so easy if they must be composed. However, this must be done only once per relational schema, and it is useful as a documentation of the semantics of the schema. On the other hand, some kind of mapping is needed in all approaches.

We have shown that from these mappings, the other semantic components of a data integration system (mediated schema, source descriptions) can in principle be automatically generated. This is the main advantage of the UO-based approach, which confirms that indeed the UO solves, or at least greatly alleviates, the semantic integration problem.

There are some aspects of the semantic integration problem that we have not discussed in this paper, such as integrity constraints, local values, local completeness or query expressions, but we believe that they would not significantly change the overall conclusion.

## References

An Y., Borgida A., Mylopoulos J. (2006) Discovering the Semantics of Relational Tables Through Mappings In: Journal on Data Semantics VII Spaccapietra S. (ed.) Springer Berlin Heidelberg, pp. 1–32 https://doi.org/10.1007/11890591_1

Arens Y., Chee C. Y., Hsu C.-N., Knoblock C. A. (1993) Retrieving and Integrating Data from Multiple Information Sources. In: International Journal of Cooperative Information Systems 02(02), pp. 127–158

Arens Y., Knoblock C. A., Shen W.-M. (1996) Query reformulation for dynamic information integration. In: Journal of Intelligent Information Systems 6(2), pp. 99–130

Batini C., Ceri S., Navathe S. B. (1992) Conceptual Database Design: An Entity-relationship Approach. Benjamin-Cummings Publishing Co., Inc.

Collet C., Huhns M. N., Shen W. M. (1991) Resource integration using a large knowledge base in Carnot. In: Computer 24(12), pp. 55–62

Doan A., Halevy A., Ives Z. (2012) Principles of Data Integration. Morgan Kaufmann, Burlington

Fowler M. (1997) Analysis Patterns: Reusable Object Models. Addison-Wesley

Grüninger M., Uschold M. (2004) Ontologies and Semantics for Seamless Connectivity. In: SIGMOD Record 33, pp. 58–64

Huhns M. N., Jacobs N., Ksiezyk T., Shen W.-M., Singh M. P., Cannata P. E. (1993) Integrating enterprise information models in Carnot. In: Proceedings International Conference on Intelligent and Cooperative Information Systems, pp. 32–42

Lenzerini M. (2002) Data Integration: A Theoretical Perspective. In: Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems,PODS '02. ACM, pp. 233–246 http://doi.acm.org/10.1145/543613.543644

Lockemann P., Mayr H., Weil W., Wohlleber W. (1979) Data Abstractions for Database Systems. In: ACM Transactions Database Systems 4, pp. 60–75

Madnick S. E. (1996) Are we moving toward an information superhighway or a Tower of Babel? The challenge of large-scale semantic heterogeneity. In: Proceedings of the Twelfth International Conference on Data Engineering, pp. 2–8

Mena E., Illarramendi A., Kashyap V., Sheth A. P. (2000) OBSERVER: An Approach for Query Processing in Global Information Systems Based on Interoperation Across Pre-Existing Ontologies. In: Distributed and Parallel Databases 8(2), pp. 223–271 http://doi.acm.org/10.1023/A:1008741824956

Michael J., Mayr H. C. (2017) Intuitive Understanding of a Modeling Language. In: Proceedings of the Australasian Computer Science Week Multiconference, ACSW '17. ACM, Geelong, Australia, 35:1–35:10 http://doi.acm.org/10.1145/3014812.3014849

Miller G. A. (1995) WordNet: A Lexical Database for English. In: Commun. ACM 38(11), pp. 39–41 http://doi.acm.org/10.1145/219717.219748

Olivé A. (2007) Conceptual modeling of information systems. Springer-Verlag Berlin Heidelberg http://doi.acm.org/10.1007/978-3-540-39390-0

Olivé A. (2017) The Universal Ontology: A Vision for Conceptual Modeling and the Semantic Web (Invited Paper) In: Conceptual Modeling: 36th International Conference (ER 2017) Mayr H. C., Guizzardi G., Ma H., Pastor O. (eds.) Springer International Publishing, pp. 1–17 https://doi.org/10.1007/978-3-319-69904-2_1

Park J., Ram S. (2004) Information Systems Interoperability: What Lies Beneath? In: ACM Transactions on Information Systems 22(4), pp. 595–632 http://doi.acm.org/10.1145/1028099.1028103

SAP (2016) Identifiers. https://help.sap.com/saphelp%5C_ewm94/helpdata/en/48/d0060005ae154ee10000000a421937/frameset.htm

UN/CEFACT (2009) Core Components Data Type Catalogue Version 3.0.. https://www.unece.org/fileadmin/DAM/cefact/codesfortrade/CCTS/CCTS-DataTypeCatalogueVersion3p0.pdf

Uschold M. (2000) Creating, Integrating and Maintaining Local and Global Ontologies. In: Proceedings of the First Workshop on Ontology Learning (OL-2000) in conjunction with the 14th European Conference on Artificial Intelligence (ECAI-2000)

# Representing Imprecise Time Intervals in OWL 2

Elisabeth Métais[*,a], Fatma Ghorbel[b], Fayçal Hamdi[a], Nebrasse Ellouze[b], Noura Herradi[a], Assia Soukane[c]

[a] CEDRIC Laboratory, Conservatoire National des Arts et Métiers, Paris, France
[b] MIRACL Laboratory, University of Sfax, Sfax, Tunisia
[c] Ecole Centrale d'Electronique, Paris, France

Abstract. *Representing and reasoning on imprecise temporal information is a common requirement in the field of Semantic Web. Many works exist to represent and reason on precise temporal information in OWL; however, to the best of our knowledge, none of these works is devoted to imprecise temporal time intervals. To address this problem, we propose two approaches: a crisp-based approach and a fuzzy-based approach. (1) The first approach uses only crisp standards and tools and is modelled in OWL 2. We extend the 4D-fluents model, with new crisp components, to represent imprecise time intervals and qualitative crisp interval relations. Then, we extend the Allen's interval algebra to compare imprecise time intervals in a crisp way and inferences are done via a set of SWRL rules. (2) The second approach is based on fuzzy sets theory and fuzzy tools and is modelled in Fuzzy-OWL 2. The 4D-fluents approach is extended, with new fuzzy components, in order to represent imprecise time intervals and qualitative fuzzy interval relations. The Allen's interval algebra is extended in order to compare imprecise time intervals in a fuzzy gradual personalized way. Inferences are done via a set of Mamdani IF-THEN rules.*

Keywords. Imprecise Time Interval • Linked Data • OWL • Allen's Interval Algebra • 4D-Fluents

## 1 Introduction

In the Semantic Web field, representing and reasoning on imprecise temporal information is a common requirement. Indeed, temporal information given by users is often imprecise. For instance, if they give the information "Alexandre was married to Nicole by 1981 to late 90" two measures of imprecision are involved. On the one hand, the information "by 1981" is imprecise in the sense that it could mean approximately from 1980 to 1982; on the other hand, the information "late 90" is imprecise in the sense that it could mean, with an increasingly possibility, from 1995 to 2000. When an event is characterized by a gradual beginning and/or ending, it is usual to represent the corresponding time span as an imprecise time interval.

* Corresponding author.
E-mail. elisabeth.metais@cnam.fr

In OWL, many works have been proposed to represent and reason on precise temporal information; however, to the best of our knowledge, there is no work devoted to represent and reason on imprecise temporal time intervals. In this paper, we answer this problem with two approaches, one using a crisp environment, the other one using a fuzzy environment.

The first approach involves only crisp[1] standards and tools. To represent imprecise time intervals in OWL 2, we extend the so called 4D-fluents model (Welty and Fikes 2006) which is a formalism to model crisp quantitative temporal information and the evolution of temporal concepts in OWL. This model is extended in two ways: (1) It

[1] The word "crisp" designates "precise", in opposite to "fuzzy" in the context of fuzzy sets theory. An ontology is either "crisp" (i.e., a "classic" ontology) or "fuzzy".

is enhanced with new crisp components for modelling imprecise time intervals. (2) It is enhanced with qualitative temporal expressions representing crisp relations between imprecise temporal intervals. To reason on imprecise time intervals, we extend the Allen's interval algebra (Allen 1983) which is the most used and known formalisms for reasoning about crisp time intervals. We generalize Allen's relationships to handle imprecise time intervals with a crisp view. The resulting crisp temporal interval relations are inferred from the introduced imprecise time intervals using a set of SWRL rules (Horrocks et al. 2004), in OWL 2.

The second approach is based on fuzzy sets theory and fuzzy tools. It is based on Fuzzy-OWL 2 (Bobillo and Straccia 2011) which is an extension of OWL 2 that deals with fuzzy information. To represent imprecise time intervals in Fuzzy-OWL 2, we extend the 4D-fluents model in two ways: (1) It is enhanced with new fuzzy components to be able to model imprecise time intervals. (2) It is enhanced with qualitative temporal expressions representing fuzzy relations between imprecise temporal intervals. To reason on imprecise time intervals, we extend Allen's work to compare imprecise time intervals in a fuzzy gradual personalized way. Our Allen's extension introduces gradual fuzzy interval relations e.g., "long before". It is personalized in the sense that it is not limited to a given number of interval relations. It is possible to determinate the level of precision that should be in a given context. For instance, the classic Allen relation "before" may be generalized in $N$ interval relations, where "before(1)" means "just before" and gradually the time gap between the two imprecise intervals increases until "before(N)" which means "long before". The resulting fuzzy interval relations are inferred from the introduced imprecise time intervals using the FuzzyDL reasoner (Bobillo and Straccia 2008), via a set of Mamdani IF-THEN rules, in Fuzzy-OWL 2.

The current paper is organized as follows: Section 2 is devoted to present some preliminary concepts and related work in the field of temporal information representation in OWL and reasoning on time intervals. In Section 3, we introduce our crisp-based approach for representing and reasoning on imprecise time intervals. In Section 4, we introduce our fuzzy-based approach for representing and reasoning on imprecise time intervals. Section 5 draws conclusions and future research directions.

## 2 Preliminaries and Related Work

In this section, we introduce some preliminary concepts and related work in the field of temporal information representation in OWL and reasoning on time intervals.

### 2.1 Representing Temporal Information in OWL

Five main approaches are proposed to represent time information in OWL: Temporal Description Logics (Artale and Franconi 2000), Versioning (Klein and Fensel 2001), N-ary relations (Noy and Rector 2006) and 4D-fluents (Welty and Fikes 2006). All these approaches represent only crisp temporal information in OWL. Temporal Description Logics extend the standard description logics with additional temporal constructs e.g., "sometime in the future". N-ary relations approach represents an N-ary relation using an additional object. The N-ary relation is represented as two properties each related with the new object. The two objects are related to each other with an N-ary relation. Reification is "a general purpose technique for representing N-ary relations using a language such as OWL that permits only binary relations" (Batsakis and Petrakis 2011). Versioning approach is described as "the ability to handle changes in ontologies by creating and managing different variants of it" (Klein and Fensel 2001). When an ontology is modified, a new version is created to represent the temporal evolution of the ontology. 4D-fluents approach represents temporal information and the evolution of the last ones in OWL. Concepts varying in time are represented as 4-dimensional objects with the 4th dimension being the temporal dimension.

Based on the present related work, we choose the 4D-fluents approach. Indeed, compared to

related work, it minimizes the problem of data redundancy as the changes occur only on the temporal parts and keeping therefore the static part unchanged. It also maintains full OWL expressiveness and reasoning support (Batsakis and Petrakis 2011). We extend this approach in two ways. (1) It is extended with crisp components to represent imprecise time intervals and crisp interval relations in OWL 2 (Section 3). (2) It is extended with fuzzy components to represent imprecise time intervals and fuzzy interval relations in Fuzzy-OWL 2 (Section 4).

### 2.2 Allen's Interval Algebra

(Allen 1983) has proposed 13 mutually exclusive primitive relations that may hold between two precise time intervals. Their semantics is illustrated in Table 1. Let $I = [I^-, I^+]$ and $J = [J^-, J^+]$ two time intervals; where $I^-$ (respectively $J^-$) is the beginning time-step of the event and $I^+$ (respectively $J^-$) is the ending.

A number of works fuzzify Allen's temporal interval relations. We classify these works into (1) works focusing on fuzzifying Allen's interval algebra to compare precise time intervals and (2) works focusing on fuzzifying Allen's interval algebra to compare imprecise time intervals.

Three approaches have been proposed to fuzzify Allen's interval algebra in order to compare precise time intervals: (Guesgen et al. 1994), (Dubois and Prade 1989) and (Badaloni and Giacomin 2006). (Guesgen et al. 1994) propose fuzzy Allen relations viewed as fuzzy sets of ordinary Allen relationship taking into account a neighbourhood structure, a notion originally introduced in (Freksa 1992). (Dubois and Prade 1989) represent a time interval as a pair of possibility distributions that define the possible values of the endpoints of the crisp interval. Using possibility theory, the possibility and necessity of each of the interval relations can then be calculated. This approach also allows modelling imprecise relations such as "long before". (Badaloni and Giacomin 2006) propose a fuzzy extension of Allen's work, called IAfuz where degrees of preference are associated to each relation between two precise time intervals.

Four approaches have been proposed to fuzzify Allen's interval algebra to compare imprecise time intervals: (Nagypál and Motik 2003), (Ohlbach 2004), Schockaert08 and (Gammoudi et al. 2017). (Nagypál and Motik 2003) propose a temporal model based on fuzzy sets to extend Allen relations with imprecise time intervals. The authors introduce a set of auxiliary operators on intervals and define fuzzy counterparts of these operators. The compositions of these relations are not studied by the authors. (Ohlbach 2004) proposes an approach to handle some gradual temporal relations as "more or less finishes". However, this work cannot take into account gradual temporal relations such as "long before". Furthermore, many of the symmetry, reflexivity, and transitivity properties of the original temporal interval relations are lost in this approach; thus it is not suitable for temporal reasoning. (Schockaert and Cock 2008) propose a generalization of Allen's relations with precise and imprecise time intervals. This approach allows handling classical temporal relations, as well as other imprecise relations. Interval relations are defined according to two fuzzy operators comparing two time instants: "long before" and "occurs before or at approximately the same time". (Gammoudi et al. 2017) generalize the definitions of the 13 Allen's classic interval relations to make them applicable to fuzzy intervals in two ways (conjunctive and disjunctive). Gradual temporal interval relations are not taken into account.

## 3 A Crisp-Based Approach for Representing and Reasoning on Imprecise Time Intervals

In this section, we propose a crisp-based approach to represent and reason on imprecise time intervals. This solution is entirely based on crisp standards and tools. We extend the 4D-fluents model to represent imprecise time intervals and their crisp relationships in OWL 2. To reason on imprecise time intervals, we extend the Allen's interval algebra in a crisp way. In OWL 2, inferences are done via a set of SWRL rules.

*Table 1: Allen's temporal interval relations (I: ——— , J: ═══).*

| Relation | Inverse | Relations between interval bounds | Illustration |
|---|---|---|---|
| $Before(I,J)$ | $After(I,J)$ | $I^+ < J^-$ | |
| $Meets(I,J)$ | $MetBy(I,J)$ | $I^+ = J^-$ | |
| $Overlaps(I,J)$ | $OverlappedBy(I,J)$ | $(I^- < J^-) \wedge (I^+ > J^-) \wedge (I^+ < J^+)$ | |
| $Starts(I,J)$ | $StartedBy(I,J)$ | $(I^- = J^-) \wedge (I^+ < J^+)$ | |
| $During(I,J)$ | $Contains(I,J)$ | $(I^- > J^-) \wedge (I^+ < J^+)$ | |
| $Ends(I,J)$ | $EndedBy(I,J)$ | $(I^- > J^-) \wedge (I^+ = J^+)$ | |
| $Equal(I,J)$ | $Equal(I,J)$ | $(I^- = J^-) \wedge (I^+ = J^+)$ | |

### 3.1 Representing Imprecise Time Intervals and Crisp Qualitative Interval Relations in OWL 2

In the crisp-based solution, we now represent each imprecise interval bound of the time interval as a disjunctive ascending set. Let $I = [I^-, I^+]$ be an imprecise time interval; where $I^- = I^{-(1)} \ldots I^{-(N)}$ and $I^+ = I^{+(1)} \ldots I^{+(N)}$. For instance, if we have the information "Alexandre was started his PhD study in 1975 and he was graduated around 1980" the imprecise time interval representing this period is $[1975, \{1978 \ldots 1982\}]$. This means that his PhD studies end in 1978 or 1979 or 1980 or 1981 or 1982. The classic 4D-fluents model introduces two crisp classes "TimeSlice" and "TimeInterval" and four crisp properties "tsTimeSliceOf", "tsTimeInterval", "hasBegining" and "hasEnd". The class "TimeSlice" is the domain class for entities representing temporal parts (i.e., "time slices"). The property "tsTimeSliceOf" connects an instance of class "TimeSlice" with an entity. The property "tsTimeInterval" connects an instance of class "TimeSlice" with an instance of class "TimeInterval". The instance of class "TimeInterval" is related with two temporal instants that specify its starting and ending points using, respectively, the "hasBegining" and "hasEnd" properties. Figure 1 illustrates the use of the 4D-fluents model to represent the following example: "Alexandre was started his PhD study in 1975 and he was graduated in 1978".

We extend the original 4D-fluents model in the following way. We add four crisp datatype properties "HasBeginningFrom", "HasBeginningTo", "HasEndFrom", and "HasEndTo" to the class "TimeInterval". Let $I = [I^-, I^+]$ be an imprecise time interval; where $I^- = I^{-(1)} \ldots I^{-(N)}$ and $I^+ = I^{+(1)} \ldots I^{+(N)}$. "HasBeginningFrom" has the range $I^{-(1)}$. "HasBeginningTo" has the range $I^{-(N)}$. "HasEndFrom" has the range $I^{+(1)}$. "HasEndTo" has the range $I^{+(N)}$. The 4D-fluents model is also enhanced with crisp qualitative temporal interval relations that may hold between imprecise time intervals. This is implemented by introducing temporal relationships, called "RelationIntervals", as a crisp object property between two instances of the class "TimeInterval". Figure 2 represents the extended 4D-fluents model in OWL 2.

We can see in Figure 3 an instantiation of the extended 4D-fluents model in OWL 2. On this example, we consider the following information: "Alexandre was married to Nicole just after he was graduated with a PhD. Alexandre was graduated with a PhD in 1980. Their marriage lasts 15 years. Alexandre was remarried to Béatrice since about 10 years and they were divorced in 2016". Let $I = [I^-, I^+]$ and $J = [J^-, J^+]$ be two imprecise time intervals representing, respectively, the duration of the marriage of Alexandre with Nicole and the one with Béatrice. Assume that $I^- = \{1980 \ldots 1983\}$,

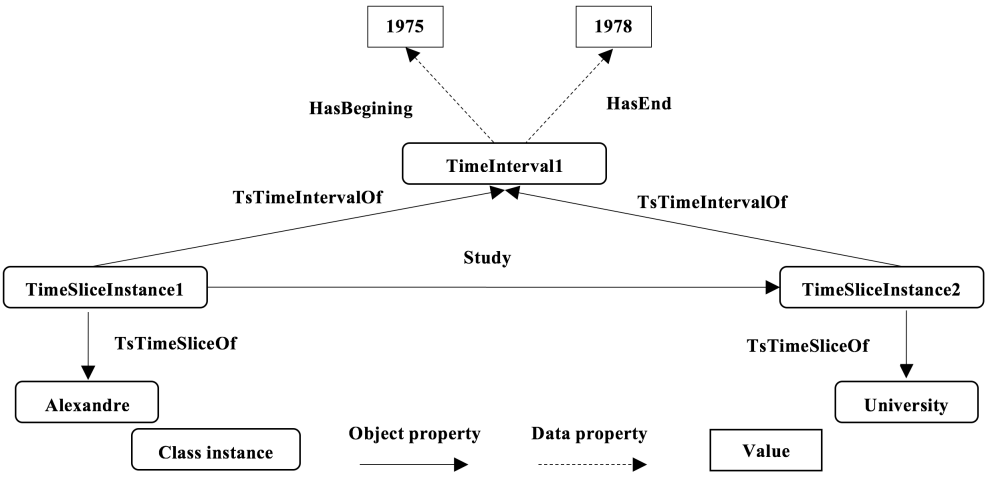



*Figure 1: An instantiation of the classic the 4D-fluents model.*

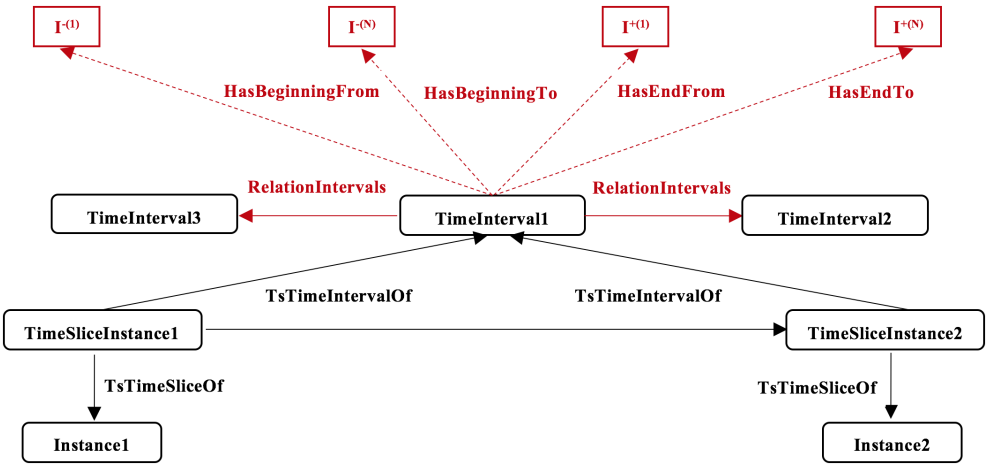



*Figure 2: The extended 4D-fluents model in OWL 2.*

$I^+ = \{1995 \ldots 1998\}$, $J^- = \{2006 \ldots 2008\}$ and $J^+ = 2016$.

### 3.2 A Crisp-Based Reasoning on Imprecise Time Intervals in OWL 2

We have redefined 13 Allen's interval, in a crisp way, to compare imprecise time intervals i.e., the resulting interval relations upon imprecise time intervals are crisp. Let $I = [I^-, I^+]$ and $J = [J^-, J^+]$ two imprecise time intervals; where $I^- = I^{-(1)} \ldots I^{-(N)}$, $I^+ = I^{+(1)} \ldots I^{+(N)}$, $J^- = J^{-(1)} \ldots J^{-(N)}$ and $J^+ = J^{+(1)} \ldots J^{+(N)}$. For instance, the crisp interval relation "before (I, J)" is redefined as: $\forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^- : I^{+(i)} < J^{-(j)}$ This means that the most recent time instant of $I^+(I^{+(N)})$ ought to Precede the oldest time instant of $J^-(J^{-(1)})$: $I^{+(N)} < J^{-(1)}$ In the similar way, we define the other temporal interval relations. In Table 2, we define 13 crisp temporal interval relations upon the two imprecise time intervals $I$ and $J$.

In order to apply our crisp extension of Allen's work in OWL 2, we propose a set of SWRL rules that infer the temporal interval relations from the introduced imprecise time intervals which are represented using the extended 4D-fluents model in OWL2. For each temporal interval relation, we associate a SWRL rule. Reasoners that support DL-safe rules (i.e., rules that apply only on named

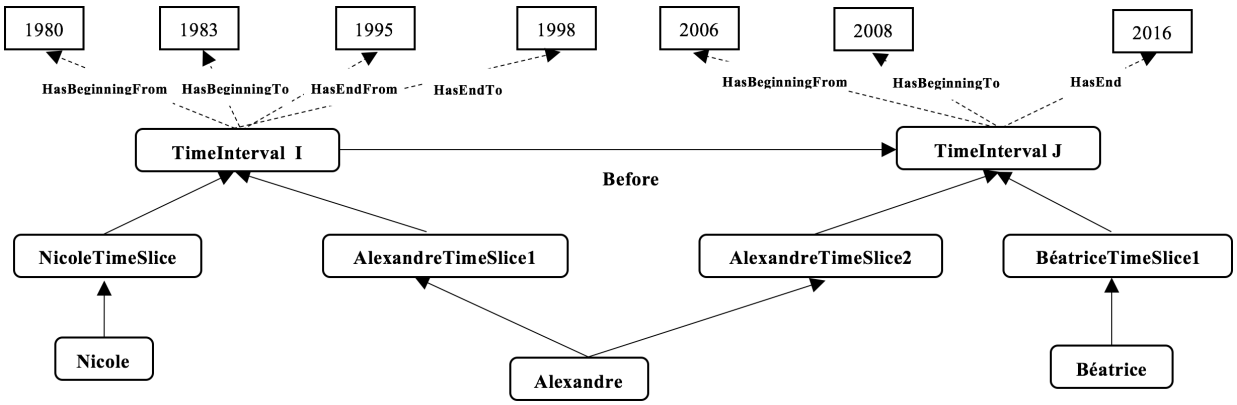



*Figure 3: An instantiation of the extended 4D-fluents model in OWL 2.*

individuals in the knowledge base) such as Pellet (Sirin et al. 2007) can support our approach. For instance, the SWRL rule to infer the "Meet (I, J)" relation is the following:

$$TimeInterval(I) \wedge TimeInterval(J) \wedge HasEndFrom(I, a) \wedge HasBeginningFrom(J, b) \wedge Equals(a, b) \wedge HasEndTo(I, c) \wedge HasBeginningTo(J, d) \wedge Equals(c, d) \rightarrow Meet(I, J)$$

## 4 A Fuzzy-Based Approach for Representing and Reasoning on Imprecise Time Intervals

In this section, we propose a fuzzy-based approach to represent and reason on imprecise time intervals. This approach is based on a fuzzy environment. We extend the 4D-fluents model to represent imprecise time intervals and their relationships in Fuzzy-OWL 2. To reason on imprecise time intervals, we extend the Allen's interval algebra in a fuzzy gradual personalized way. We infer the resulting fuzzy interval relations in Fuzzy-OWL 2 using a set of Mamdani IF-THEN rules.

### 4.1 Representing Imprecise Time Intervals and Fuzzy Qualitative Interval Relations in Fuzzy-OWL 2

In the fuzzy-based solution, we now represent the imprecise beginning interval bound as a fuzzy set which has the L-function MF and the ending interval bound as a fuzzy set which has the R-function membership function (MF). Let $I = [I^-, I^+]$ be an imprecise time interval. We represent the binging bound $I^-$ as a fuzzy set which has the L-function MF ($A = I^{-(1)}$ and $B = I^{-(N)}$). We represent the ending bound $I^+$ as a fuzzy set which has the R-function MF ($A = I^{+(1)}$ and $B = I^{+(N)}$). For instance, if we have the information "Alexandre was starting his PhD study in 1973 and was graduated in late 80", the beginning bound is crisp. The ending bound is imprecise and it is represented by L-function MF ($A = 1976$ and $B = 1980$). For the rest of the paper, we use the MFs shown in Figure 4 (Zadeh 1975).

We extend the original 4D-fluents model to represent imprecise time intervals in the following way. We add two fuzzy datatype properties "FuzzyHasBegining" and "FuzzyHasEnd" to the class "TimeInterval". "FuzzyHasBegining" has the L-function MF ($A = I^{-(1)}$ and $B = I^{-(N)}$). "FuzzyHasEnd" has the R-function MF ($A = I^{+(1)}$ and $B = I^{+(N)}$). The 4D-fluents approach is also enhanced with qualitative temporal relations that may hold between imprecise time intervals. We introduce the "FuzzyRelationIntervals", as a fuzzy object property between two instances of the class "TimeInterval". "FuzzyRelationIntervals" represent fuzzy qualitative temporal relations. "FuzzyRelationIntervals" has the L-function MF ($A = 0$ and $B = 1$). Figure 5 represents our extended 4D-fluents model in Fuzzy-OWL 2.

*Table 2: Crisp temporal interval relations upon imprecise time intervals.*

| Relation | Inverse | Interpretation | Relations between interval bounds |
|---|---|---|---|
| $Before(I,J)$ | $After(I,J)$ | $\forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^- : (I^{+(i)} < J^{-(j)})$ | $I^{+(N)} < J^{-(1)}$ |
| $Meets(I,J)$ | $MetBy(I,J)$ | $\forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^- : (I^{+(i)} = J^{-(j)})$ | $(I^{+(1)} = J^{-(1)}) \wedge (I^{+(N)} = J^{-(N)})$ |
| $Overlaps(I,J)$ | $OverlappedBy(I,J)$ | $\forall I^{-(i)} \in I^-, \forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^-, \forall J^{+(j)} \in J^+ : (I^{-(i)} < J^{-(j)}) \wedge (J^{-(j)} < I^{+(i)}) \wedge (I^{+(i)} < J^{+(j)})$ | $(I^{-(N)} < J^{-(1)}) \wedge (J^{-(N)} < I^{+(1)}) \wedge (I^{+(N)} < J^{+(1)})$ |
| $Starts(I,J)$ | $StartedBy(I,J)$ | $\forall I^{-(i)} \in I^-, \forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^-, \forall J^{+(j)} \in J^+ : (I^{-(i)} = J^{-(j)}) \wedge (I^{+(i)} < J^{+(j)})$ | $(I^{-(1)} = J^{-(1)}) \wedge (I^{-(N)} = J^{-(N)}) \wedge (I^{+(N)} < J^{+(1)})$ |
| $During(I,J)$ | $Contains(I,J)$ | $\forall I^{-(i)} \in I^-, \forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^-, \forall J^{+(i)} \in J^+ : (J^{-(j)} < I^{-(i)}) \wedge (I^{+(i)} < J^{+(i)})$ | $(J^{-(N)} < I^{-(1)}) \wedge (I^{+(N)} < J^{+(1)})$ |
| $Ends(I,J)$ | $EndedBy(I,J)$ | $\forall I^{-(i)} \in I^-, \forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^-, \forall J^{+(j)} \in J^+ : (I^{-(i)} < J^{-(j)}) \wedge (I^{+(i)} = J^{+(j)})$ | $(J^{-(N)} < I^{-(1)}) \wedge (I^{+(1)} = J^{+(1)}) \wedge (I^{+(N)} = J^{+(N)})$ |
| $Equal(I,J)$ | $Equal(I,J)$ | $\forall I^{-(i)} \in I^-, \forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^-, \forall J^{+(j)} \in J^+ : (I^{-(i)} = J^{-(j)}) \wedge (I^{+(i)} = J^{+(j)})$ | $(I^{-(1)} = J^{-(1)}) \wedge (I^{-(N)} = J^{-(N)}) \wedge (I^{+(1)} = J^{+(1)}) \wedge (I^{+(N)} = J^{+(N)})$ |

We can see in Figure 6 an instantiation of the extended 4D-fluents model in Fuzzy-OWL 2. On this example, we consider the following information: "Alexandre was married to Nicole just after he was graduated with a PhD. Alexandre was graduated with a PhD in 1980. Their marriage lasts 15 years. Alexandre was remarried to Béatrice since about 10 years and they were divorced in 2016". Let $I = [I^-, I^+]$ and $J = [J^-, J^+]$ be two imprecise time intervals representing, respectively, the duration of the marriage of Alexandre with Nicole and the one with Béatrice. $I^-$ is represented with the fuzzy datatype property "FuzzyHasBegining" which has the L-function MF ($A$ = 1980 and $B$ = 1983). $I^+$ is represented with the fuzzy datatype property "FuzzyHasEnd" which has the R-function MF ($A$ = 1995 and $B$ = 1998). $J^-$ is represented with the fuzzy datatype property "FuzzyHasBegining" which has the L-function MF ($A$ = 2005 and $B$ = 2007). $J^+$ is represented with the crisp datatype property "HasEnd" which has the value "2016".

## 4.2 A Fuzzy-Based Reasoning on Imprecise Time Intervals in Fuzzy OWL 2

We propose a set of fuzzy gradual personalized comparators that may hold between two time instants. Based on these operators, we present our fuzzy gradual personalized extension of Allen's work. Then, we infer, in Fuzzy OWL 2, the resulting temporal interval relations via a set of Mamdani IF-THEN rules using the fuzzy reasoner FuzzyDL. We generalize the crisp time instants comparators "Follow", "Precede" and "Same", introduced in (Vilain and Kautz 1986). Let $\alpha$ and $\beta$ two parameters allowing the definition of the membership function of the following comparators ($\in ]0, +\infty[$); $N$ is the number of slices; $T_1$ and $T_2$ are two time instants; we define the following comparators (illustrated in Figure 7):

- $\{Follow_{(1)}^{(\alpha,\beta)}(T_1,T_2) \ldots Follow_{(N)}^{(\alpha,\beta)}(T_1,T_2)\}$ are a generalization of the crisp time instants relation "Follows". $Follow_{(1)}^{(\alpha,\beta)}(T_1,T_2)$ means that $T_1$ is "just after or approximately at the same time" $T_2$ w.r.t. $(\alpha,\beta)$ and gradually the time gap between $T_1$

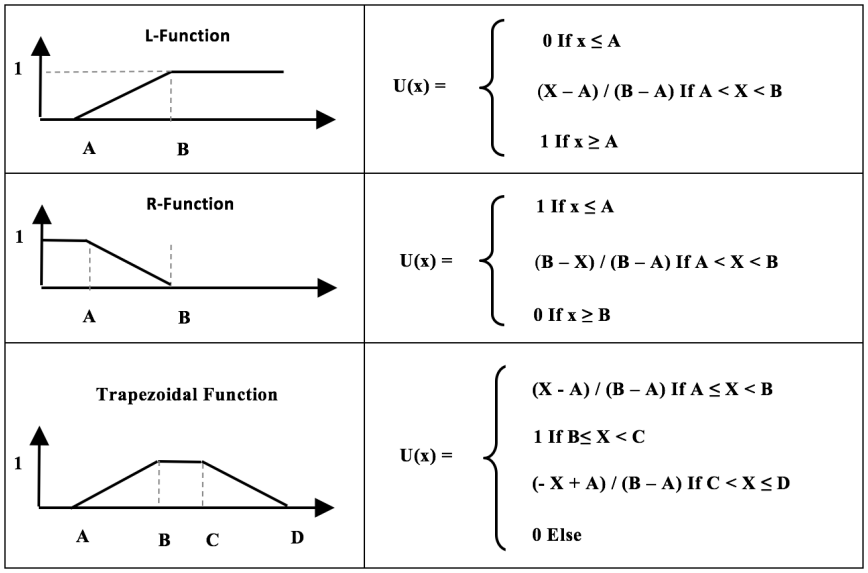



*Figure 4: R-Function, L-Function and Trapezoidal MFs (Zadeh 1975).*

and $T_2$ increases until $Follow_{(N)}^{(\alpha,\beta)}(T_1, T_2)$ which means that $T_1$ is "long after" $T_2$ w.r.t. $(\alpha, \beta)$. $N$ is set by the expert domain. $\{Follow_{(1)}^{(\alpha,\beta)}(T_1, T_2) \ldots Follow_{(N)}^{(\alpha,\beta)}(T_1, T_2)\}$ are defined as fuzzy sets. $Follow_{(1)}^{(\alpha,\beta)}(T_1, T_2)$ has R-Function MF which has as parameters $A = \alpha$ and $B = (\alpha + \beta)$. All comparators $\{Follow_{(2)}^{(\alpha,\beta)}(T_1, T_2) \ldots Follow_{(N-1)}^{(\alpha,\beta)}(T_1, T_2)\}$ have trapezoidal MF which has as parameters $A = ((K-1)\alpha)$ and $B = ((K-1)\alpha + (K-1)\beta)$, $C = (K\alpha + (K-1)\beta)$ and $D = (K\alpha + K\beta)$; where $2 \leq K \leq N-1$. $Follow_{(N)}^{(\alpha,\beta)}(T_1, T_2)$ has L-Function MF which has as parameters $A = ((N-1)\alpha + (N-1)\beta)$ and $B = ((N-1)\alpha + (N-1)\beta)$;

- $\{Precede_{(1)}^{(\alpha,\beta)}(T_1, T_2) \ldots Precede_{(N)}^{(\alpha,\beta)}(T_1, T_2)\}$ are a generalization of the crisp time instants relation "Precede". $Precede_{(1)}^{(\alpha,\beta)}(T_1, T_2)$ means that $T_1$ is "just before or approximately at the same time" $T_2$ w.r.t. $(\alpha, \beta)$ and gradually the time gap between $T_1$ and $T_2$ increases until $Precede_{(N)}^{(\alpha,\beta)}(T_1, T_2)$ which means that $T_1$ is "long before" $T_2$ w.r.t. $(\alpha, \beta)$. $N$ is set by the expert domain. $\{Precede_{(1)}^{(\alpha,\beta)}(T_1, T_2) \ldots Precede_{(N)}^{(\alpha,\beta)}(T_1, T_2)\}$ are defined as fuzzy sets. $Precede_{(i)}^{(\alpha,\beta)}(T_1, T_2)$ is defined as:

$$Precede_{(i)}^{(\alpha,\beta)}(T_1, T_2) = 1 - Follow_{(i)}^{(\alpha,\beta)}(T_1, T_2)$$

- We define the comparator $Same^{(\alpha,\beta)}$ which is a generalization of the crisp time instants relation "Same". $Same^{(\alpha,\beta)}(T_1, T_2)$ means that $T_1$ is "approximately at the same time" $T_2$ w.r.t. $(\alpha, \beta)$. It is defined as:

$$Same^{(\alpha,\beta)}(T_1, T_2) = Min(Follow_{(1)}^{(\alpha,\beta)}(T_1, T_2), Precede_{(1)}^{(\alpha,\beta)}(T_1, T_2))$$

Then, we extend Allen's work to compare imprecise time intervals with a fuzzy gradual personalized view. We provide a way to model gradual, linguistic-like description of temporal interval relations. Compared to related work, our work is not limited to a given number of imprecise relations. It is possible to determinate the level of precision that should be in a given context. For instance, the classic Allen relation "before" may be generalized in $N$ imprecise relations, where "$Before_{(1)}^{(\alpha,\beta)}(I, J)$" means that $I$ is "just before" $J$ w.r.t. $(\alpha, \beta)$ and gradually the time gap between $I$ and $J$ increases until "$Before_{(N)}^{(\alpha,\beta)}(I, J)$" which

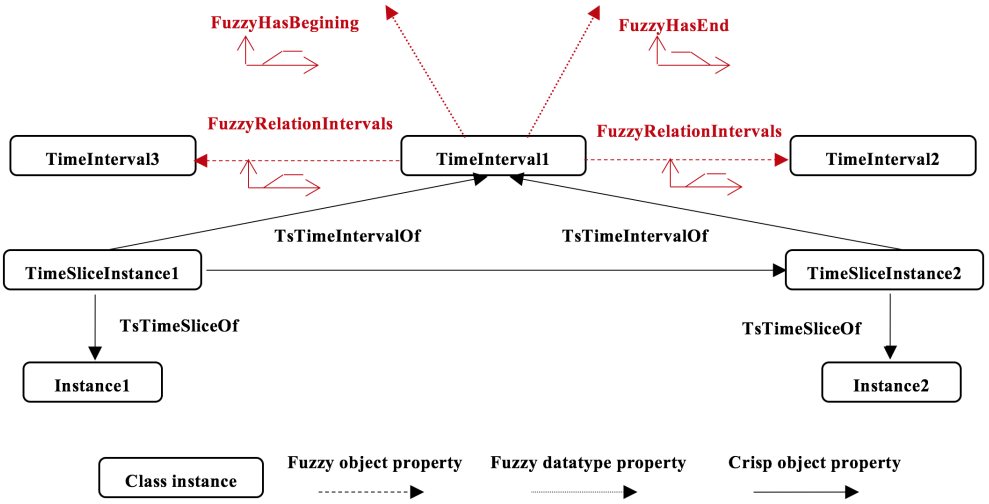



*Figure 5: The extended 4D-fluents model in Fuzzy-OWL 2.*

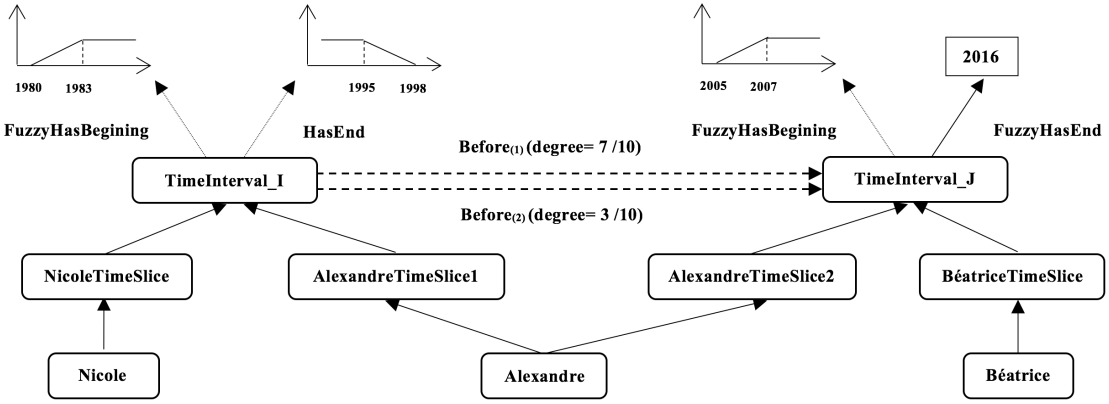



*Figure 6: An instantiation of the extended 4D-fluents model in Fuzzy-OWL 2.*

means that $I$ is long before $J$ w.r.t. $(\alpha, \beta)$. The definition of our fuzzy interval relations is based on the fuzzy gradual personalized time instants compactors. Let $I = [I^-, I^+]$ and $J = [J^-, J^+]$ two imprecise time intervals; where $I^-$ has the L-function MF ($A = I^{-(1)}$ and $B = I^{-(N)}$); $I^+$ is a fuzzy set which has the R-function MF ($A = I^{+(1)}$ and $B = I^{+(N)}$); $J^-$ is a fuzzy set which has the L-function MF ($A = J^{-(1)}$ and $B = J^{-(N)}$); $J^+$ is a fuzzy set which has the R-function MF ($A = J^{+(1)}$ and $B = J^{+(N)}$). For instance, the fuzzy interval relation "$Before^{(\alpha,\beta)}_{(1)}(I, J)$" is defined as:

$$\forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^- : \\ Precede^{(\alpha,\beta)}_{(1)}(I^{+(i)}, J^{-(j)})$$

This means that the most recent time instant of $I^+(I^{+(N)})$ ought to proceed the oldest time instant of $J^-(J^{-(1)})$:

$$Precede^{(\alpha,\beta)}_{(1)}(I{+}(N), J{-}(1))$$

In the similar way, we define the others temporal interval relations, as shown in Table 3.

Finally, we have implemented our fuzzy gradual personalized extension of Allen's work in Fuzzy-OWL 2. We use the ontology editor PROTEGE version 4.3 and the fuzzy reasoner FuzzyDL. We propose a set of Mamdani IF-THEN rules to infer the temporal interval relations from the introduced imprecise time intervals which are represented using the extended 4D-fluents model in

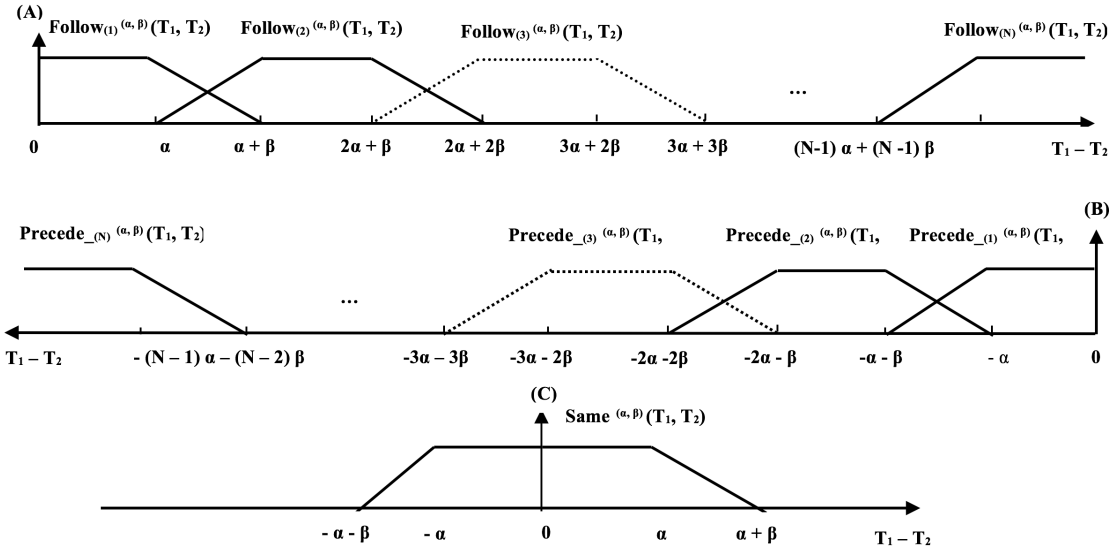



*Figure 7: Fuzzy gradual personalized time instants comparators. (A) Fuzzy sets of* $\{Follow_{(1)}^{(\alpha,\beta)}(T_1,T_2)\ldots Follow_{(N)}^{(\alpha,\beta)}(T_1,T_2)\}$. *(B) Fuzzy sets of* $\{Precede_{(1)}^{(\alpha,\beta)}(T_1,T_2)\ldots Precede_{(N)}^{(\alpha,\beta)}(T_1,T_2)\}$. *(C) Fuzzy set of* $Same^{(\alpha,\beta)}(T_1,T_2)$.

*Table 3: Fuzzy gradual personalized temporal interval relations upon imprecise time intervals.*

| Relation | Inverse | Relations between bounds | Definition |
|---|---|---|---|
| $Before_{(K)}^{(\alpha,\beta)}(I,J)$ | $After_{(K)}^{(\alpha,\beta)}(I,J)$ | $\forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^- : (I^{+(i)} < J^{-(j)})$ | $Precede_{(K)}^{(\alpha,\beta)}(I^{+(N)}, J^{-(1)})$ |
| $Meets^{(\alpha,\beta)}(I,J)$ | $MetBy^{(\alpha,\beta)}(I,J)$ | $\forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^- : (I^{+(i)} = J^{-(j)})$ | $Min(Same^{(\alpha,\beta)}(I^{+(1)}, J^{-(1)}) \wedge Same^{(\alpha,\beta)}(I^{+(N)}, J^{-(N)}))$ |
| $Overlaps_{(K)}^{(\alpha,\beta)}(I,J)$ | $OverlappedBy_{(K)}^{(\alpha,\beta)}(I,J)$ | $\forall I^{-(i)} \in I^-, \forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^-, \forall J^{+(j)} \in J^+ : (I^{-(i)} < J^{-(j)}) \wedge (J^{-(j)} < I^{+(i)}) \wedge (I^{+(i)} < J^{+(j)})$ | $Min(Precede_{(K)}^{(\alpha,\beta)}(I^{-(N)}, J^{-(1)}) \wedge Precede_(K)^{(\alpha,\beta)}(J^{-(N)}, I^{+(1)}) \wedge Precede_{(K)}^{(\alpha,\beta)}(I^{+(N)}, J^{+(1)}))$ |
| $Starts_{(K)}^{(\alpha,\beta)}(I,J)$ | $StartedBy_{(K)}^{(\alpha,\beta)}(I,J)$ | $\forall I^{-(i)} \in I^-, \forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^-, \forall J^{+(j)} \in J^+ : (I^{-(i)} = J^{-(j)}) \wedge (I^{+(i)} < J^{+(j)})$ | $Min(Same^{(\alpha,\beta)}(I^{-(1)}, J^{-(1)}) \wedge Same^{(\alpha,\beta)}(I^{-(N)}, J^{-(N)}) \wedge Precede_{(K)}^{(\alpha,\beta)}(I^{+(N)}, J^{+(1)}))$ |
| $During_{(K)}^{(\alpha,\beta)}(I,J)$ | $Contains_{(K)}^{(\alpha,\beta)}(I,J)$ | $\forall I^{-(i)} \in I^-, \forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^-, \forall J^{+(i)} \in J^+ : (J^{-(j)} < I^{-(i)}) \wedge (I^{+(i)} < J^{+(i)})$ | $Min(Precede_{(K)}^{(\alpha,\beta)}(J^{-(N)}, I^{-(1)}) \wedge Precede_{(K)}^{(\alpha,\beta)}(I^{+(N)}, J^{+(1)}))$ |
| $Ends_{(K)}^{(\alpha,\beta)}(I,J)$ | $EndedBy_{(K)}^{(\alpha,\beta)}(I,J)$ | $\forall I^{-(i)} \in I^-, \forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^-, \forall J^{+(j)} \in J^+ : (I^{-(i)} < J^{-(j)}) \wedge (I^{+(i)} = J^{+(j)})$ | $Min(Precede_{(K)}^{(\alpha,\beta)}(J^{-(N)}, I^{-(1)}) \wedge Same^{(\alpha,\beta)}(I^{+(1)}, J^{+(1)}) \wedge Same^{(\alpha,\beta)}(I^{+(N)}, J^{+(N)}))$ |
| $Equal^{(\alpha,\beta)}(I,J)$ | $Equal^{(\alpha,\beta)}(I,J)$ | $\forall I^{-(i)} \in I^-, \forall I^{+(i)} \in I^+, \forall J^{-(j)} \in J^-, \forall J^{+(j)} \in J^+ : (I^{-(i)} = J^{-(j)}) \wedge (I^{+(i)} = J^{+(j)})$ | $Min(Same^{(\alpha,\beta)}(I^{-(1)}, J^{-(1)}) \wedge Same^{(\alpha,\beta)}(I^{-(N)}, J^{-(N)}) \wedge Same^{(\alpha,\beta)}(I^{+(1)}, J^{+(1)}) \wedge Same^{(\alpha,\beta)}(I^{+(N)}, J^{+(N)}))$ |

Fuzzy-OWL2. For each temporal interval relation, we associate a Mamdani IF-THEN rule. For instance, the Mamdani IF-THEN rule to infer the "$Overlaps_{(1)}^{(\alpha,\beta)}(I, J)$" relation is the following:

*(define-concept Rule0 (g-and (some* $Precede_{(1/1)}$ *Fulfilled) (some* $Precede_{(1/2)}$ *Fulfilled) Fulfilled) (some* $Precede_{(1/3)}$ *Fulfilled) (some* $Overlaps_{(1)}$ *True))) // Fuzzy rule*

We define three input fuzzy variables, named "Precede(1/1)", "Precede(1/2)" and "Precede(1/3)", which have the same MF than that of "$Precede_{(1)}^{(\alpha,\beta)}$". We define one output variable "Overlaps(1)" which has the same membership than that of the fuzzy object property "FuzzyRelationIntervals". "Precede(1/1)", "Precede(1/2)" and "Precede(1/3)" are instantiated with, respectively, $(I^{-(N)} - J^{-(1)})$, $(J^{-(N)} - I^{+(1)})$ and $(I^{+(N)} - J^{+(1)})$.

## 5 Conclusion

In this paper, we proposed two approaches to represent and reason on imprecise time intervals in OWL: a crisp-based approach and a fuzzy-based approach. (1) The crisp-based approach is based only on crisp environment. We extended the 4D-fluents model to represent imprecise time intervals and crisp interval relations in OWL 2. To reason on imprecise time intervals, we extended the Allen's interval algebra in a crisp way. Inferences are done via a set of SWRL rules. (2) The fuzzy-based approach is entirely based only on fuzzy environment. We extended the 4D-fluents model to represent imprecise time intervals and fuzzy interval relations in Fuzzy-OWL 2. To reason on imprecise time intervals, we extend the Allen's interval algebra in a fuzzy gradual personalized way. We infer the resulting fuzzy interval relations in Fuzzy-OWL 2 using a set of Mamdani IF-THEN rules.

Concerning the choice between these two approaches (the crisp-based one or the fuzzy-based one), as a fuzzy ontology is an extension of crisp ontology, researchers may choose any of our two approaches for introducing imprecise interval management in their knowledge bases whatever these latter are crisp or fuzzy. However our fuzzy-based approach allows more functionalities, in particular it is suitable to represent and reason on gradual interval relations such as "middle before" or "approximately at the same time". Hence, in the case of manipulating a fuzzy knowledge base, we encourage researchers to choose the fuzzy-based approach to model and reason on imprecise time intervals. The main interest of the crisp-based approach is that this solution can be implemented with classical crisp tools and that the programmers are not obliged to learn technologies related to fuzzy ontology. Considering that crisp tools and models are more mature and better support scaling, the crisp-based approach is more suitable for marketed products.

The works presented in this paper have been tested in two projects, having in common to manage life logging data: (1) In the VIVA[2] project, we aim to design the Captain Memo memory prosthesis (Herradi et al. 2015; Métais et al. 2015) for Alzheimer Disease patients. Among other functionalities, this prosthesis manages a knowledge base on the patient's family tree, using an OWL ontology. Imprecise inputs are especially numerous when given by an Alzheimer Disease patient. Furthermore, dates are often given in reference to other dates or events. Thus, we have been using the "fuzzy" solution reported in this paper. One interesting point in this solution is to deal with a personalized slicing of the person's life in order to sort the different events. (2) The QUALHIS[3] project aims to allow Middle Ages specialized historians to deal with prosopographical data bases storing Middle age academic's career histories. Data come from various archives among Europe and data about a same person are very difficult to align. Representing and ordering imprecise time interval is required to redraw the careers across the different European universities who hosted the

[2] http://viva.cnam.fr/

[3] http://www.univ-paris1.fr/fileadmin/Axe_de_recherche_PIREH/aap2017-mastodons-Lamasse.pdf

person. We preferred the "crisp" solution in order to favour the integration within the existing crisp ontologies and tools, and to ease the management by Historians.

## Acknowledgments

We are very grateful to Pr. Heinrich Mayr for all the energy and the enthusiasm he insufflates in the international research in Computer Science and we are highly impressed by his seminal role in numerous domains of Conceptual Modeling. Furthermore he also never lost his kindness and humanity in spite of being one of the greatest researchers in Europe.

## References

Allen J. F. (1983) Maintaining Knowledge about Temporal Intervals. In: Commun. ACM 26(11), pp. 832–843 http://doi.acm.org/10.1145/182.358434

Artale A., Franconi E. (2000) A survey of temporal extensions of description logics. In: Ann. Math. Artif. Intell. 30(1-4), pp. 171–210 https://doi.org/10.1023/A:1016636131405

Badaloni S., Giacomin M. (2006) The algebra IA$^{fuz}$: a framework for qualitative fuzzy temporal reasoning. In: Artif. Intell. 170(10), pp. 872–908 https://doi.org/10.1016/j.artint.2006.04.001

Batsakis S., Petrakis E. G. M. (2011) SOWL: A Framework for Handling Spatio-temporal Information in OWL 2.0. In: Bassiliades N., Governatori G., Paschke A. (eds.) Rule-Based Reasoning, Programming, and Applications - 5th International Symposium, RuleML 2011 - Europe, Barcelona, Spain, July 19-21, 2011. Proceedings. Lecture Notes in Computer Science Vol. 6826. Springer, pp. 242–249 https://doi.org/10.1007/978-3-642-22546-8_19

Bobillo F., Straccia U. (2008) fuzzyDL: An expressive fuzzy description logic reasoner. In: FUZZ-IEEE 2008, IEEE International Conference on Fuzzy Systems, Hong Kong, China, 1-6 June, 2008, Proceedings. IEEE, pp. 923–930 https://doi.org/10.1109/FUZZY.2008.4630480

Bobillo F., Straccia U. (2011) Fuzzy ontology representation using OWL 2. In: Int. J. Approx. Reasoning 52(7), pp. 1073–1094 https://doi.org/10.1016/j.ijar.2011.05.003

Dubois D., Prade H. (1989) Processing fuzzy temporal knowledge. In: IEEE Trans. Systems, Man, and Cybernetics 19(4), pp. 729–744 https://doi.org/10.1109/21.35337

Freksa C. (1992) Temporal Reasoning Based on Semi-Intervals. In: Artif. Intell. 54(1), pp. 199–227 https://doi.org/10.1016/0004-3702(92)90090-K

Gammoudi A., Hadjali A., Yaghlane B. B. (2017) Fuzz-TIME: an intelligent system for managing fuzzy temporal information. In: Int. J. Intelligent Computing and Cybernetics 10(2), pp. 200–222 https://doi.org/10.1108/IJICC-09-2016-0036

Guesgen H. W., Hertzberg J., Philpott A. (1994) Towards implementing fuzzy Allen relations. In: Proceedings of the ECAI-94 Workshop on Spatial and Temporal Reasoning, pp. 49–55

Herradi N., Hamdi F., Métais E., Ghorbel F., Soukane A. (2015) PersonLink: An Ontology Representing Family Relationships for the CAPTAIN MEMO Memory Prosthesis. In: Jeusfeld M. A., Karlapalem K. (eds.) Advances in Conceptual Modeling - ER 2015 Workshops, AHA, CMS, EMoV, MoBiD, MORE-BI, MReBA, QMMQ, and SCME Stockholm, Sweden, October 19-22, 2015, Proceedings. Lecture Notes in Computer Science Vol. 9382. Springer, pp. 3–13 https://doi.org/10.1007/978-3-319-25747-1_1

Horrocks I., Patel-Schneider P. F., Boley H., Tabet S., Grosofand B., Dean M. (2004) SWRL: A Semantic Web Rule Language Combining OWL and RuleML. http://www.w3.org/Submission/SWRL/

Klein M. C. A., Fensel D. (2001) Ontology versioning on the Semantic Web. In: Cruz I. F., Decker S., Euzenat J., McGuinness D. L. (eds.) Proceedings of SWWS'01, The first Semantic Web Working Symposium, Stanford University, California, USA, July 30 - August 1, 2001, pp. 75–91 http:

//www.semanticweb.org/SWWS/program/full/paper56.pdf

Métais E., Ghorbel F., Herradi N., Hamdi F., Lammari N., Nakache D., Ellouze N., Gargouri F., Soukane A. (2015) Memory Prosthesis. In: Non-pharmacological Therapies in Dementia 3(2), pp. 177–180

Nagypál G., Motik B. (2003) A Fuzzy Model for Representing Uncertain, Subjective, and Vague Temporal Knowledge in Ontologies. In: Meersman R., Tari Z., Schmidt D. C. (eds.) On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE - OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Lecture Notes in Computer Science Vol. 2888. Springer, pp. 906–923 https://doi.org/10.1007/978-3-540-39964-3_57

Noy N., Rector A. (2006) Defining N-ary Relations on the Semantic Web. W3C Note. W3C. Last Access: http://www.w3.org/TR/2006/NOTE-swbp-n-aryRelations-20060412/

Ohlbach H. J. (2004) Relations Between Fuzzy Time Intervals. In: 11th International Symposium on Temporal Representation and Reasoning (TIME 2004), 1-3 July 2004, Tatihou Island, Normandie, France. IEEE Computer Society, pp. 44–51 https://doi.org/10.1109/TIME.2004.1314418

Schockaert S., Cock M. D. (2008) Temporal reasoning about fuzzy intervals. In: Artificial Intelligence 172(8), pp. 1158–1193 http://www.sciencedirect.com/science/article/pii/S0004370208000039

Sirin E., Parsia B., Grau B. C., Kalyanpur A., Katz Y. (2007) Pellet: A practical OWL-DL reasoner. In: J. Web Sem. 5(2), pp. 51–53 https://doi.org/10.1016/j.websem.2007.03.004

Vilain M. B., Kautz H. A. (1986) Constraint Propagation Algorithms for Temporal Reasoning. In: Kehler T. (ed.) Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, PA, August 11-15, 1986. Volume 1: Science.. Morgan Kaufmann, pp. 377–382 http://www.aaai.org/Library/AAAI/1986/aaai86-063.php

Welty C. A., Fikes R. (2006) A Reusable Ontology for Fluents in OWL. In: Bennett B., Fellbaum C. (eds.) Formal Ontology in Information Systems, Proceedings of the Fourth International Conference, FOIS 2006, Baltimore, Maryland, USA, November 9-11, 2006. Frontiers in Artificial Intelligence and Applications Vol. 150. IOS Press, pp. 226–236 http://www.booksonline.iospress.nl/Content/View.aspx?piid=2209

Zadeh L. A. (1975) The concept of a linguistic variable and its application to approximate reasoning - II. In: Inf. Sci. 8(4), pp. 301–357 https://doi.org/10.1016/0020-0255(75)90046-8

# Ontological Document Reading

## An Experience Report

David W. Embley*,[a], Stephen W. Liddle[b], Deryle W. Lonsdale[b], Scott N. Woodfield[b]

[a] FamilySearch International, Lehi, Utah, USA
[b] Brigham Young University, Provo, Utah, USA

Abstract. *Ontological document reading is defined as automatically and appropriately populating a conceptual model representing an ontological conceptualization of some fragment of the real world. Appropriately populating the conceptualization involves not only extracting the information with respect to the declared object and relationship sets of the conceptual model but also involves checking the extracted information for real-world constraint violations, standardizing the data, and inferring the unwritten information that a document author intended to convey. Appropriately populating an ontology may, in addition, require adjustments to the ontology itself. This approach to document reading is presented in terms of an effort to build a system to extract the genealogical information in family history books. The status of the reading system is reported. Also explained is how the generated results can be imported into and thus contribute to the construction of a large repository of world-wide family interrelationships. The reading system's potential use for constructing similar knowledge repositories in other domains is foreshadowed.*

Keywords. Document Reading • Information Extraction • Conceptual Modeling • Ontology Conceptualization • Extraction Ontology

## 1 Introduction

*Ontology* is the philosophical study of the nature of being, existence, or reality. *An ontology* is a shared, commonly agreed upon conceptualization of a domain of interest (Gruber 1993). An ontology can be represented as a conceptual model, which names and defines the types, properties, and interrelationships of the objects that exist in a particular domain (Dillon et al. 2008; Guizzardi and Halpin 2008).

*Ontological document reading*, in its simplest form, consists of automatically populating a conceptual model with object and relationship instance facts extracted from a document's text. More completely described, *ontological document reading* consists of four interrelated tasks:

1. Populate multiple interrelated conceptual models with facts stated in a document and integrate the extracted information into a unified whole.
2. Check the extracted facts with respect to ontological constraints supplied as part of the conceptual models.
3. Infer facts called for in the conceptual models but not explicitly stated in the document.
4. Construct and augment conceptual models to capture document facts not conceptualized in a given collection of conceptual models.

As an example of ontological document reading, consider reading the document in Figure 1 with respect to the ontological conceptualization in Figure 2:

* Corresponding author.
E-mail. embley@cs.byu.edu

241213. Mary Eliza Warner, b. 1826, dau. of Samuel Selden Warner and Azubah Tully; m. 1850, Joel M. Gloyd (who was connected with Chief Justice Waite's family).

243311. Abigail Huntington Lathrop (widow), Boonton, N. J., b. 1810, dau. of Mary Ely and Gerard Lathrop; m. 1835, Donald McKenzie, West Indies, who was b. 1812, d. 1839.

(The widow is unable to give the names of her husband's parents.) Their children:

1. Mary Ely, b. 1836, d. 1859.
2. Gerard Lathrop, b. 1838.

243312. William Gerard Lathrop, Boonton, N. J., b. 1812, d. 1882, son of Mary Ely and Gerard Lathrop; m. 1837, Charlotte Brackett Jennings, New York City, who was b. 1818, dau. of Nathan Tilestone Jennings and Maria Miller. Their children:

1. Maria Jennings, b. 1838, d. 1840.
2. William Gerard, b. 1840.
3. Donald McKenzie, b. 1840, d. 1843. } Twins.
4. Anna Margaretta, b. 1843.
5. Anna Catherine, b. 1845.

243314. Charles Christopher Lathrop, N. Y. City, b. 1817, d. 1865, son of Mary Ely and Gerard Lathrop; m. 1856, Mary Augusta Andruss, 992 Broad St., Newark, N. J., who was b. 1825, dau. of Judge Caleb Halstead Andruss and Emma Sutherland Goble. Mrs. Lathrop died at her home, 992 Broad St., Newark, N. J., Friday morning, Nov. 4, 1898. The funeral services were held at her residence on Monday, Nov. 7, 1898, at half-past two o'clock P. M. Their children:

1. Charles Halstead, b. 1857, d. 1861.
2. William Gerard, b. 1858, d. 1861.
3. Theodore Andruss, b. 1860.
4. Emma Goble, b. 1862.

Miss Emma Goble Lathrop, official historian of the New York Chapter of the Daughters of the American Revolution, is one of the youngest members to hold office, but one whose intelligence and capability qualify her for such distinction. Miss Lathrop is not without experience; in her present home and native city, Newark, N. J., she has filled the positions of secretary and treasurer to the Girls' Friendly Society for nine years, secretary and president of the Woman's Auxiliary of Trinity Church Parish, treasurer of the St. Catherine's Guild of St. Barnabas Hospital, and manager of several of Newark's charitable institutions which her grandparents were instrumental in founding. Miss Lathrop traces her lineage back through many generations of famous progenitors on both sides. Her maternal ancestors were among the early settlers of New Jersey, among them John Ogden, who received patent in 1664 for the purchase of Elizabethtown, and who in 1673 was

*Figure 1: Page 419 of The Ely Ancestry (Vanderpoel 1902)*

1. *Ontology Population*
   Extraction results should include the following facts: *Person(Mary Eliza Warner) was born on BirthDate(1826). Person(Mary Eliza Warner) married Spouse(Joel M. Gloyd) on MarriageDate(1850) at BirthPlace(unknown). Child(Maria Jennings) is a child of Person(William Gerard Lathrop). Child(Maria Jennings) is a child of Person(Charlotte Brackett Jennings).*
2. *Constraint Checking*
   Human readers should, and usually do, implicitly check extracted facts against ontological reality. Automated extraction tools, with no intuition of their own, often make extraction mistakes that are wildly unreasonable. For example, because the OCR system makes a mistake—interpreting "1" in Theodore Andruss's birth year as "i", which actually happens in the OCR of the document in Figure 1—an automated extractor can extract *Person(Theodore Andruss) was born on BirthDate(860)*. By itself this extracted fact is reasonable, but not in the context of other extracted facts: *Child(Theodore Andruss) is a child of Person(Mary Augusta Andruss)* and *Person(Mary August Andruss) was born on BirthDate(1825)*. These extracted facts along with the constraint that a child cannot be born before its mother imply that at least one of them is incorrect.
3. *Fact Inference*
   Document authors typically do not explicitly state all the facts they wish to convey. Readers of the page in Figure 1, for example, should infer (a) the gender of a person (e.g. *Person(Mary Eliza Warner) has Gender(Female)*); (b) roles such as father, mother, wife, husband (e.g. *Mother(Abigail Huntington Lathrop)* because she is female and has children); (c) full married names of female spouses based on cultural traditions (e.g. *Person(Mary Eliza Warner) has InferredMarriedName(Mary Eliza Warner Gloyd)*; and (d) full birth names (e.g. *Person(Maria Jennings) has InferredBirthName(Maria Jennings Lathrop)*). Readers should also be able to determine that some name instances refer to the same person. In the last family on the page in Figure 1, for example, *Person(Mrs. Lathrop)* is the same as *Person(Mary Augusta Andruss)*.
4. *Ontology Construction and Augmentation*
   Using a named entity recognizer, we can create a simple ontology with two linked concepts, the entity *E* and its name *EName* (e.g. *Location has LocationName*). Note that this conceptualization is similar to the conceptualization *Person has Name* in Figure 2. Furthermore, just as the appearance of a person's name instantiates a *Person* object and links it to the name, the appearance of a location name instantiates a *Location* object and links it to the location's name. From the page in Figure 1 three of the extractable location facts are *Location(West Indies)*, *Location(Boonton, N. J.)*, and *Location(N. Y. City)*. A reader can infer from the text that there is an association respectively between these locations and *Person(Donald McKenzie)*, *Person(William Gerard Lathrop)*, and *Person(Charles Christopher Lathrop)*. Natural language processing systems can too. Hence, having determined that there is an implied association between *Person* and *Location*, an automated ontology construction system can connect the two ontologies, augmenting both, on its way to constructing an ever-growing collection of ontologies that can be populated by an ontological document reading system.

A tremendous amount of academic research has contributed to the grand challenge of document understanding. Ontology, the nature of reality, and epistemology, the theory of knowledge, have been the subject of research since the days of Aristotle (Aristotle about 350BC). In modern times, several disciplines within computer science and linguistics have contributed to document understanding as indicated by the many conferences and workshops that have sprung up surrounding the topic (e.g. AAAI, ACL, DAS, EMNLP, ER, ICDAR, ICPR, IJCAI, NLDB, SIGMOD, SIGIR, SIGRAPH, and

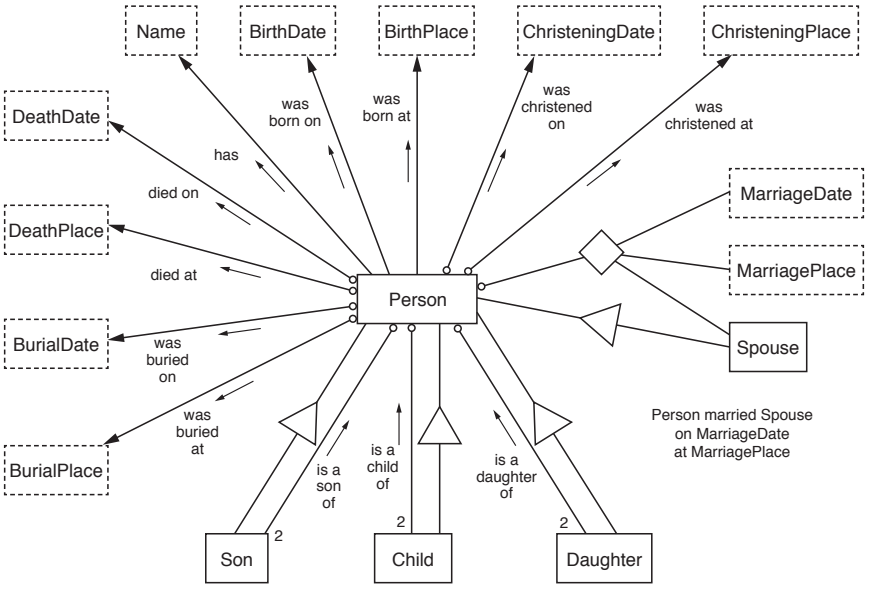



*Figure 2: A Genealogy Ontology*

TREC, among others). The grand challenge of document understanding involves much more than just document reading. It includes image processing, handwriting recognition, optical character recognition, identification of document components (e.g. figures, tables, text), determination of text reading order, and much more. Document reading, on which we focus in this experience report, is an essential component, but only a component, of the larger grand challenge.

The contributions of this report include:

1. a definition and description of ontological document reading;
2. a prototype implementation of an ontological document reading system; and
3. a real-world application of the implementation that extracts and organizes information for and contributes to the online and ever-growing *Family Tree* (FamilySearch n.d.), a public wiki-like, shared repository of interconnected family genealogies that contains more than 1.2 billion person names and associated information.

We present the details of these contributions as follows. Section 2 explains how we populate conceptual models from text. Section 3 describes our information-extraction tools and how they map the facts they extract into ontological conceptualizations. Section 4 discusses the integration of the conceptual models our extraction tools use and of the information that populates these conceptual models. Section 5 describes how we reason about extracted and inferred facts with respect to given ontological constraints. It also describes how the system can sometimes resolve invalid extraction results and can always at least point specifically to what is likely wrong, so that system users can resolve them. Section 6 explains how we infer facts of interest implied by, but not stated, in a document. It also discusses resolving object identity for multiple mentions of the same person. Section 7 explains how we can (semi)automatically construct and evolve ontological conceptualizations so that additional information can be gleaned from a document. Section 8 gives the status of our implementation and the results of some field experiments we have conducted to evaluate the effectiveness of our document reading system. It also reports on some initial contributions to *Family Tree*, and it looks to future work and application-enhancement opportunities. Section 9 summarizes the work and makes concluding remarks.

## 2 Ontological Text Extraction

Research on automated text extraction began in the information retrieval community and can be dated back at least to Salton's groundwork (Salton 1968). Researchers in other computer science disciplines soon joined in the quest to extract information from text. NLP researchers have made significant progress on named entity recognition (NER) in free running text (Nadeau and Sekine 2007) and have more recently focused on recognizing relationships among named entities (e.g. (Schone and Gehring 2016)). Researchers within the database, library/information science, document analysis, and AI communities have sought to make documents more easily searchable and to extract specified items of information. Hundreds of research papers in these various disciplines have been published in many journals and conference proceedings (Grishman 2015; Jiménez et al. 2016; Laender et al. 2002; Sarawagi 2008; Turmo et al. 2006).

For ontological document reading, extraction results must be mapped to an ontology. Early extraction systems (e.g. (Lehnert et al. 1994), (Kushmerick et al. 1997), and those surveyed in (Eikvil 1999)) labeled extracted data with category names, often called "slots." But these slot conceptualizations lacked the structure and interconnectedness of larger ontologies and the richness of their constraints and inference capabilities. Later work on linguistically grounding ontologies begins to open the door to ontological document reading (Buitelaar et al. 2009).

In our extraction work, we create ontologies as the target for information extraction and directly populate these ontologies with information extraction engines. We specify ontologies as conceptual models using the OSM conceptual modeling language (Embley et al. 1992). We augment these conceptual models with extraction rules that map document text to object and relationship instances in the conceptual model's object and relationship sets. An ontological conceptual model augmented with extraction rules is an *extraction ontology* that can read a document and populate its object and relationship sets with information gleaned from the document. We have developed an ensemble of tools for creating extraction rules and for otherwise generating mappings from text to ontological conceptualizations. These tools rely on techniques for developing rule-based expert systems and for doing natural language processing (NLP), cognitive reasoning, and machine learning.

### 2.1 Ontological Conceptualizations

Figure 3 shows an OSM conceptual model. Rectangular boxes are *object sets*—dashed if lexical and solid if non-lexical. The objects stored in lexical object sets are strings. In Figure 3 *SpouseName* and *Year* are lexical and may have strings such as "Mary Augusta Andruss" and "1826" as members of their respective object sets. Non-lexical objects are represented as numbered object identifiers—(e.g. "osmx73", because our underlying representation for **OSM** conceptual models is **X**ML).

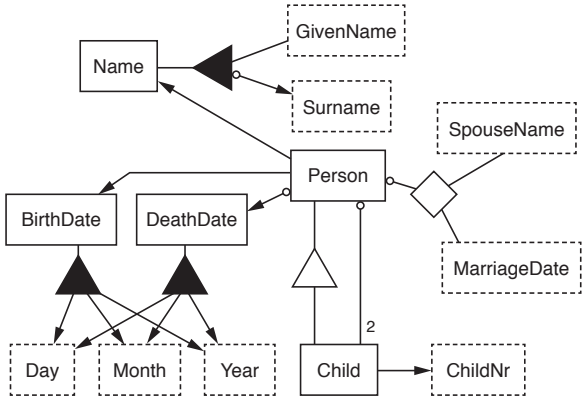



*Figure 3: Basic Ely Ontology*

Lines between object sets denote *relationship sets*; a diamond may appear in the middle of a line and usually does for three or more connecting object sets—e.g. the marriage relationship set in Figure 3. Object-set/relationship-set connections each have a *participation constraint*, a special kind of cardinality constraint (Liddle et al. 1993) that specifies the number of times an object in an object set can participate in a relationship set. The *2* in Figure 3, for example, specifies that a *Child* is related to exactly two *Person*s—the

child's parents. Graphical decorations on object-set/relationship-set connections denote common participation constraints: a small "o" for **o**ptional participation and the absence of an "o" for mandatory participation; and an arrowhead for functional participation from tail object set to head object set, which limits objects in the tail-side object set to participate at most once. In Figure 3, for example, a *Person* mandatorily has one *BirthDate* and optionally has one *DeathDate*. Also, *Person*s must have a *Name*, but they need not be married nor have children. Relationship sets have names, either explicit, as are the relationship-set names in Figure 1, or implicit, as in Figure 3. Both explicit and implicit relationship-set names include the names of the connected object sets. Explicit relationship-set names for binary relationship sets have a reading direction arrow that specifies which of the object set names comes before and which comes after the connecting verbiage in the full relationship-set name. In Figure 1, *Person was buried at BurialPlace* and *Son is a son of Person* are two of the relationship-set names. Implicit names are a space concatenation of the object-set names in any order—e.g. *Person SpouseName MarriageDate* for the marriage relationship set in Figure 3. By default, implicit names for functional binary relationship sets, may use *has* as the connecting verbiage, read from tail to head—e.g. *Person has BirthDate* in Figure 3.

A white-filled triangle denotes a *generalization/specialization* with one or more specializations connected to the base and a generalization connected to the apex. The objects in a specialization are a subset of the objects in a generalization, and all the connecting relationship sets of a generalization are inherited by its specialization(s). In Figure 3, every *Child* object is also a *Person* object and has a *Name* and *BirthDate* and may be married and may have died. A *Child* may have a *ChildNr*, but a *Person* who is not a *Child* may not have a *ChildNr*.

A black-filled triangle denotes an *aggregation* with two or more component-part object sets connected to the base and the aggregate object set connected to the apex. Aggregate and component-part object sets can independently either be lexical or non-lexical. The black-filled triangle is just a grouping of ordinary relationship sets with an implied composition whose implicit names are all *x is part of y* where *x* is a component-part object-set name and *y* is the aggregate object-set name. Decorations on the connections specify participation requirements in the aggregate. In Figure 3 *BirthDate* and *DeathDate* are each an aggregate of a *Day*, *Month*, and *Year*; and *Name* is an aggregate of one or more *GivenName*s and an optional *Surname*.

## 2.2 Extraction Ontologies

When writing the *The Ely Ancestry* (Vanderpoel 1902), the author may have conceptually had in mind the ontology in Figure 3 populated with information. The job of a document reader is to reverse the process—extract the information from the book and populate the ontology. We enable an ontology to populate itself by attaching enriched linguistic recognizers to every object set and relationship set and also to selected *ontology snippets*—coherent subcomponent views of an ontology. A reading-enabled ontology is an *extraction ontology* (Embley et al. 1999a; Embley and Zitzelberger 2010; Park 2015).

In an extraction ontology, every lexical object set has an associated *data frame* (Embley 1980). In essence, a data frame for a lexical object *L* describes the objects that may populate *L* including how to recognize object instances in a document and how these objects may behave and interact with other objects. More formally, a data frame is an abstract data type whose value set *V* has an instance recognizer that identifies lexical patterns denoting values in *V* and whose set of operations *O* includes an input operator to convert identified instances to the internal representation for *V* and an output operator to convert instances in *V* to strings, as well as applicable operations such as the Boolean operator "between" for two successive dates. Except for input and output, each operation *o* in *O* has an operator recognizer that identifies

lexical patterns as indicators that *o* applies, e.g. "is between (Date) and (Date)".

Figure 4 shows an example of a *BirthDate* data frame. From the value expression and the left and right context expressions, we form a regular-expression extraction rule by concatenating the given regular expressions and placing capture-group parentheses around the value expression. The extraction rule generated from the *YearOnly* recognizer in Figure 4 – `b[.,]?\s(\b\d{4}\b)` – extracts 17 of the 18 birth years in Figure 1 into the *BirthDate* object set in Figure 2. (Theodore Andruss's birth year has an OCR error, "i860" instead of "1860", causing his birth year to be missed.) The *MonthDayYear* expression in Figure 4 recognizes dates like "Nov. 4, 1898" in Figure 1, in which `{Month}` is a macro referring to a lexicon of month names and abbreviations. Keyword phrases like `born` and `birth` in Figure 4 are not part of the regular-expression extraction rule. Instead, if data-frame extraction rules of two or more object sets recognize the same string of characters as being possible instances of themselves, the keywords and keyword phrases help disambiguate the intended target object set for the extraction. For example, to which *Date* object set in Figure 2 does the date "Nov. 4, 1898" in Figure 1 belong? A keyword `died` found in the same sentence as the date indicates that it should be extracted into the *DeathDate* object set, which has an identical *MonthDayYear* recognizer.

```
internal representation: int // Julian date: yyyyddd
external representation:
    // YearOnly
        value expression: \b\d{4}\b
        left context expression: b[.,]?\s
        right context expression:
        keyword phrases: \bborn\ | \bbirth\
    // MonthDayYear
        value expression:
        \b{Month}\s(?:1\d|2\d|30|31|\d)[.,]?\s(?:\d{4})\b
    ...
methods:
    ageAtDeath(b:BirthDate, d:DeathDate) returns int {
        return d/1000 - b/1000
    }
    ...
```

*Figure 4: BirthDate Data Frame (partial)*

In an extraction ontology, every non-lexical object set is populated by *ontological commitment*—a relation between a language and objects postulated to exist by that language. *Person* objects in Figure 2 are instantiated by the appearance of a person's name in a document. In a data frame for a non-lexical object set *N* we specify which related lexical object instantiations also instantiate an object of *N*. The instantiation of objects in *N* also instantiates relationships between non-lexical objects and their related lexical objects. For example, extracting "Mary Eliza Warner" in Figure 1 into the *Name* object set in Figure 2 causes a new object, say *osmx103*, to be placed in the *Person* object set and the relationship *Person(osmx103) has Name(Mary Eliza Warner)* to be placed in the *Person has Name* relationship set. Because of ontological commitment, we can and often do write this relationship as *Person(Mary Eliza Warner)*.

In general, ontological commitment may be declared directly by association with one or more lexical object sets or indirectly through one or more non-lexical object sets. In Figure 3, for example, the non-lexical *Name* object set is instantiated when either of the lexical object sets *GivenName* or *Surname* is instantiated, and the instantiation of a *Person* object happens when an object is instantiated in the non-lexical *Name* object set. Objects in the non-lexical object set *Child* in Figure 3 are instantiated by inheritance when a name is known to be a child name. The object existence rule `\b\d\d?[.]\s{Person}` identifies a child in Figure 1 by a person name following the appearance of a one- or two-digit number, a period, and a space. In this object existence rule `{Person}` is a macro which devolves to the macro `{Name}` which, in turn, devolves to the regular-expression rule for either *GivenName* or *Surname*, or both.

Extraction rules for relationship sets build on data-frame-specified extraction rules for object sets. As an example, the rule `{Person}[\s\S]{1,30}?b\.\s{BirthDate}` correctly extracts from Figure 1 most of the *Person was born on BirthDate* relationships for the extraction ontology in Figure 2. (It incorrectly

assigns Emma Goble's birth year to Theodore Andruss because of the OCR error, "i860", and it assigns Mary Augusta Andruss's birth year to what it extracts as a person name in the address text between Mary's name and her birth year.) The macro references to `{Person}` and `{BirthDate}` illustrate how relationship-set extraction rules build on object-set extraction rules. All combinations of value-expressions in the data frames for each referenced object set are plugged in to create extraction rules for the relationship set.

Extending relationship-set extraction rules to span across multiple relationship sets lets us define extraction rules for a coherent subcomponent of an ontology. We call these subcomponents *ontology snippets*. As an example, we can write the ontology snippet extraction rule

```
(\d)\.\s([A-Z]\w+)\s([A-Z]\w+),\sb[,.]
\s([\d|i]\d{3})(?:,\sd\.\s(\d{4}))?\.
```

which extracts all eleven numbered children along with their birth year, and, if stated, their death year. To complete the extraction rule we must match capture groups with object sets. For the ontology in Figure 3, Capture Group *1* associates with *ChildNr*, *2* and *3* associate with *GivenName*, *4* associates with *BirthDate Year*, and *5* associates with *DeathDate Year*. Note that we can associate an ordered sequence of capture groups with a single object set, as we do here for *GivenName*. This feature also lets us capture lexical items that do not appear together such as name and surname in a phone book listing where the surname is factored to the top of the list of all names with the same surname.

## 3 Learning to Read

There are a number of ways a computer can be taught to read: (1) It can be told what the patterns in the text mean (Section 3.1). (2) It can be told how ontology-equivalent forms should be filled in (Section 3.2). (3) It can learn by example (Sections 3.3 and 3.4). (4) It can discover patterns and how they map to conceptualizations (Section 3.5). (5) It can learn from training data (Section 3.6). (6) It can learn by natural language processing (NLP) and cognitive reasoning (Section 3.7).

Before describing our extraction tools, we first observe that for historical documents we are given neither the words in the document nor the lines of text. Instead the OCR provides only the characters recognized and their bounding boxes. Reconstructing the text flow in the document can sometimes be non-trivial. Particular problems occur when the point size changes or when words are typeset with comparatively large spaces between letters to maintain right justification. The document in Figure 1 has neither of these problems, and the reconstructed text is relatively clean. Other than the OCR errors of "i" for "1" and "." for "," and vice versa which we have already mentioned or alluded to, the only error is the failure of the OCR engine to recognize "Twins" which is between the lines and the brace which spans multiple lines. Reconstructing tab stops is another matter. We usually left justify each line in the OCR, but have the option to add tab stops at the beginning of lines at what appears to be every level of indentation—four of them for the page in Figure 1.

### 3.1 FROntIER: Extraction Rule Creation with Data Frames

Figure 5 shows our FROntIER ontology workbench (Park 2015) with which we can create extraction rules by hand as explained in Section 2.2. The rectangle, diamond, and triangle icons at the top let users respectively create object sets, relationship sets and generalization/specializations of an ontology. The *Tools* menu lets users select which kind of extraction rule they wish to create. The data frame rule creation tool is currently open showing an object existence rule that has been created.

We note that extraction rule creation by machine learning is more popular in academic circles than in industry. When it comes to practical application, however, rule creation is the clear winner, especially for large vendors, with 67% of their implementations being pure rule-based systems and another 17% being a hybrid of rule-based and

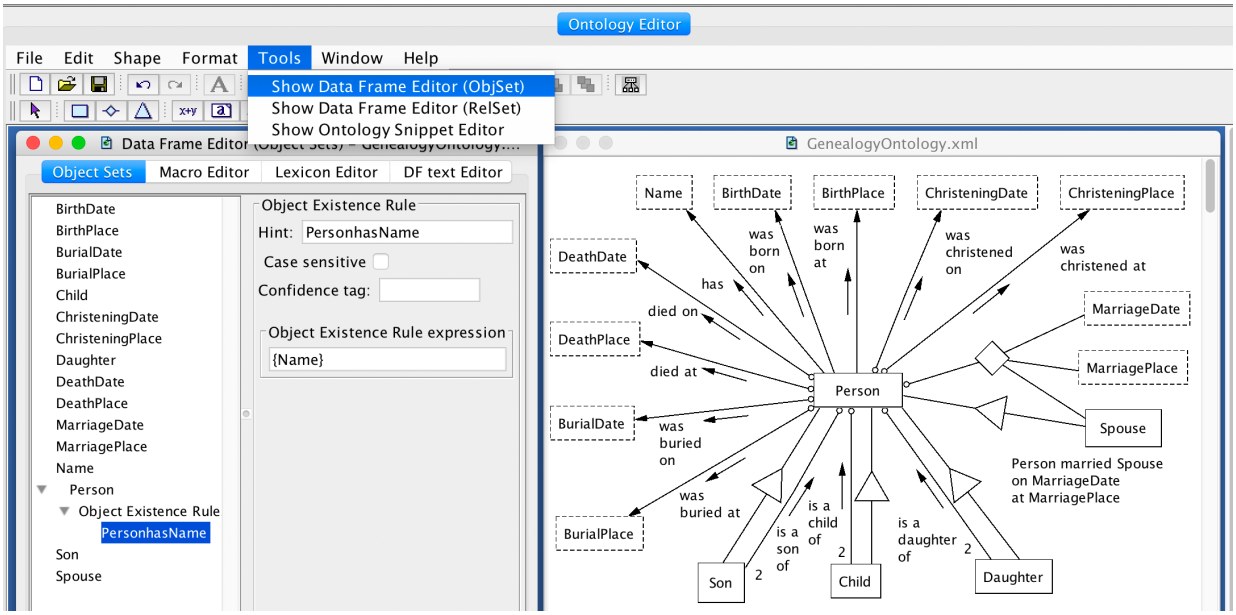



*Figure 5: Ontology Workbench*

machine-learned information extraction systems (Chiticariu et al. 2013). Five of seven of our extraction tools are based on rules. In three of the five, however, the rules are system-generated.

## 3.2 OntoES: Form-based Extraction Rule Creation

Forms can be designed so that they have a one-to-one correspondence with an ontology. Creation of such a form induces an ontology (Tao et al. 2009). In Figure 6 the form being created corresponds to the ontology in Figure 7. The form title *Person* becomes a non-lexical object set, and the single-entry form fields nested under *Person* become lexical object sets functionally dependent on *Person*. The construction menu in Figure 6 is attached to the *BirthDate* field. By clicking on *Single* we could nest lexical *Day*, *Month*, and *Year* fields under what would then be a non-lexical *BirthDate* field. We can create the ontology in Figure 8 by making the form title be *Couple* and nesting under it a single-entry form field called *Name* and a multiple-entry field which is extended to have three form fields, *SpouseName*, *MarriageDate*, and *MarriagePlace*. Similarly, we can create the ontology in Figure 9 by making the form title be *Family* and nesting under it two single-entry form fields, *Parent1* and *Parent1*, and a multiple-entry form field *Child*.

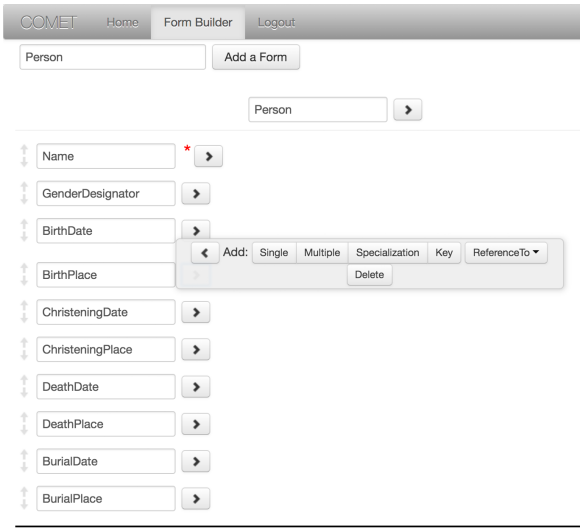



*Figure 6: Form Creation and Ontology Induction*

We can also create ontologies that have specializations. Clicking on *Specialization* in the construction menu for a field in Figure 6 nests a specialization under it. Then with the construction menu attached to the specialization, clicking on

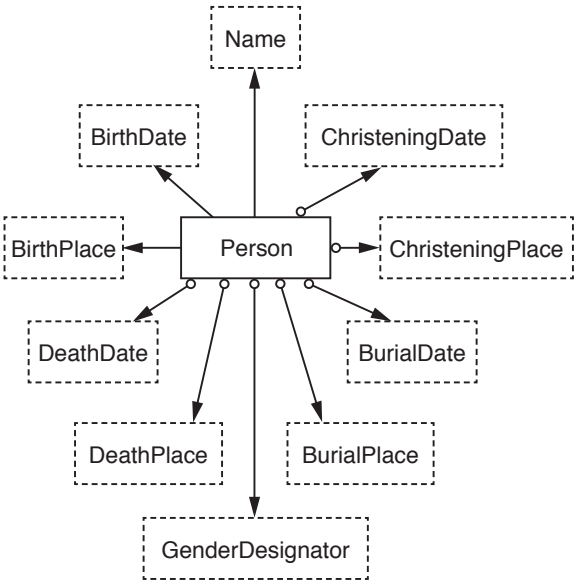



*Figure 7: Person Ontology*

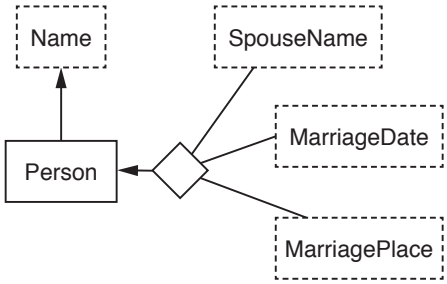



*Figure 8: Couple Ontology*

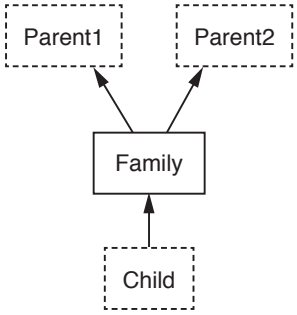



*Figure 9: Family Ontology*

*Reference To* allows a user to select any of the existing object sets to which a connecting relationship set is added. Thus, for example, we can create *Child* in Figure 3 as a specialization of *Person* and also create the relationship between *Child* and *Person*. *Key* in the construction menu is for specifying ontological commitment and designates the field to which the menu is attached as an instantiator of the non-lexical field under which it is nested. The star next to the *Name* field in Figure 6 marks *Name* as the instantiator field for the *Person* ontology in Figure 7.

Having built forms, a user *U* can "teach" the reading system how to fill them in. Clicking on the menu icon in the upper right of Figure 6, lets *U* choose a form and a page in a book and then bring up the interface in Figure 10, albeit initially with only the page and form header. *U* can then create an extraction rule by naming it and typing in a regular expression whose capture groups correspond to form fields. After specifying the match between capture groups and form fields, *U* can click on the *Test* button to test the extraction rule. Figure 10 shows the result: the form records are created and filled in, and the highlighting shows the correspondence among filled in form fields, regular-expression capture groups, and text extracted from the page. If satisfactory, *U* can save the rule along with others in an extraction ontology for the form.

### 3.3 GreenFIE: Extraction Rule Learning by Form Filling Examples

Rather than writing regular expressions to tell the computer how to fill in form records, a user can fill in a form manually. In the background, GreenFIE (Kim 2017) "watches" a user fill in a form record, generates a regular-expression extraction rule that would also have filled in the same record, generalizes the rule, executes it, and automatically fills in form records with information that matches the generalized extraction rule.

Figure 11 shows the interface of our form-filling tool, which we call COMET—**C**lick-**O**nly, or at least **M**ostly, **E**xtraction **T**ool (Embley et al. 2017). To fill in a field currently in focus (i.e. the one

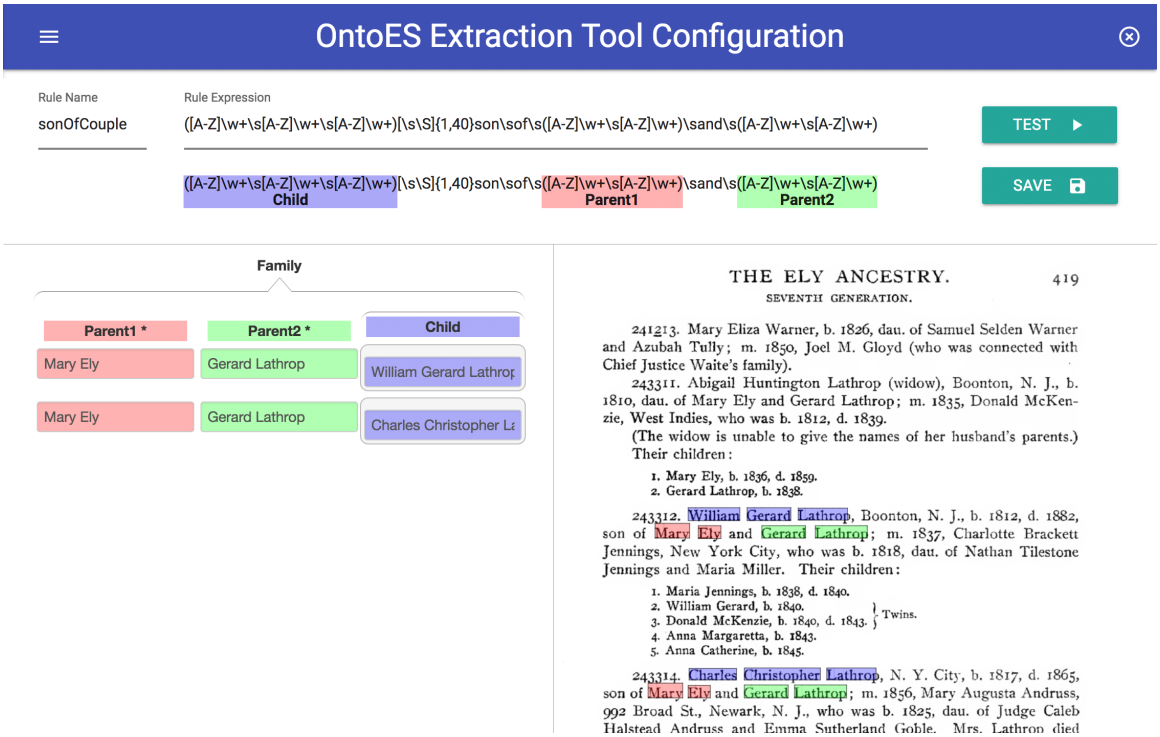



*Figure 10: OntoES Rule Editor*

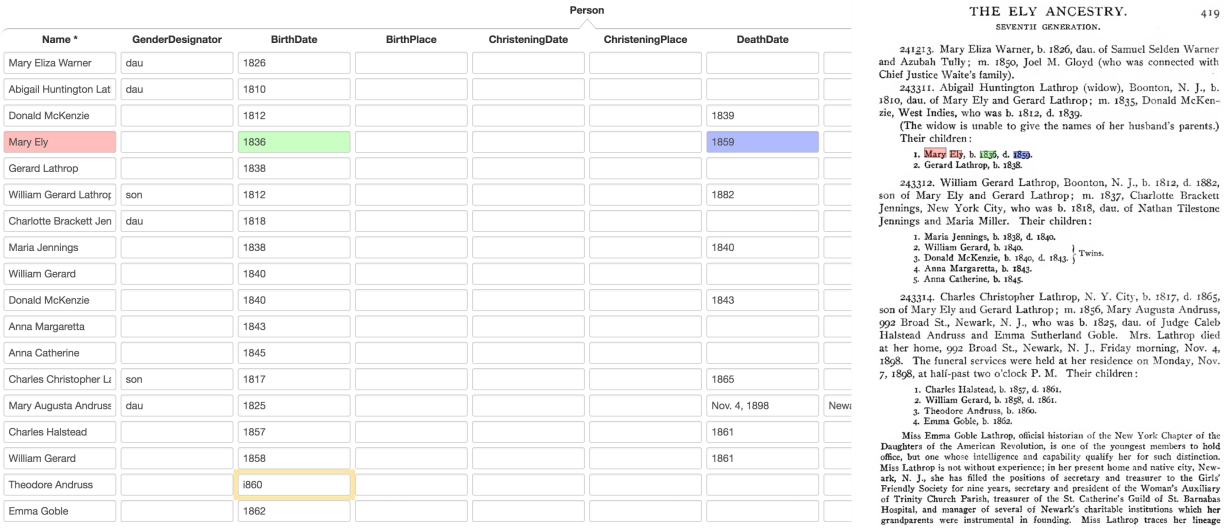



*Figure 11: GreenFIE User Interface*

with its border highlighted), a user need only click on the text string or strings in the document to be filled into the form. For historical documents, the image of a document page is superimposed over hidden OCR'd text, so that clicking on a word in the image extracts the OCR'd text of the word and enters it into the focus field. In Figure 11, for example, the field in focus is the *BirthDate* field for Theodore Andruss, and when a user clicks on "1860" in the image, the underlying OCR'd text "i860" is entered into the field. The user can correct the OCR error by double-clicking on the field and editing the text. To make it easy to see what text in the document has been extracted into which form record, a user can use a mouse to hover over a record, which then highlights each field in the record with a different color and also highlights in the document the text filled into a field with the same color. In Figure 11, the user is hovering over the record of Mary Ely who was born in 1836 and died in 1859.

The filled in form records in Figure 11 were generated with the help of GreenFIE as follows: Beginning with a single empty record, a user *U* extracted the highlighted Mary Ely information into the empty record. *U* then clicked on the *Regex* button at the end of the filled-in record. (In Figure 11, the *Regex* buttons are hidden behind the document page, but are accessible with a horizontal slider bar below the form records.) As a result, GreenFIE added the four other children in the page that also have both a birth and a death date. (After adding records, GreenFIE sorts them in page order, which is why Mary Ely's record eventually ends up as the fourth record in Figure 11.)

Next, *U* filled in a new empty record for Mary Ely's brother Gerard Lathrop, who has only a birth year, and clicked on the record's *Regex* button. For this case GreenFIE-generated the regular expression,

```
\n\d{1}[.,]\s([A-Z][a-z]+\s(?:(?:Mc
[A-Z][a-z]+)|(?:[A-Z][a-z]+)))[.,]\s
b[.,]\s(\d{4})[.,]
```

which we use here for illustration since it is the shortest of all the expressions. GreenFIE generates and generalizes using several heuristics: (1) The left context for a capture group is the text back to the first whitespace character before the word preceding the capture group. (2) The right context of a capture group consists of all immediate punctuation characters following the capture group; or if none, then `\s` is added. (3) Between capture groups, if the right of the first overlaps or abuts against the left of the second, the text between is the literal context; otherwise the left and right context is kept and a skip, `[\s\S]{`*m*`,`*n*`}`, is added where *m* and *n* are set heuristically depending on the actual number of characters being skipped. (4) Some OCR errors are anticipated, here just "." for "," and vice versa. (5) Strings of *n* digits become `\d{n}`. (6) Person names, dates, and place names are selected from a library of regular expressions depending on which expressions match the person names, dates, and place names in the given example. The regular expression here matches all the numbered children with only a birth year except Theodore's record which has the mentioned OCR error. It also matches all the numbered children with both a birth year and a death year, but GreenFIE keeps only subsuming records and thus does not generate new form records for them.

Continuing, user *U* next filled in a record for Mary Eliza Warner and her birth date, but the generated regular expression found no other matching text pattern. Similarly, generated records for both Abigail Huntington Lathrop and the West Indies Donald McKenzie yielded no other form records. The generated expression for William Gerard Lathrop of Boonton, N. J., however, does match the record for Charles Christopher Lathrop of N. Y. City, and the expression generated for Charlotte Brackett Jennings does match the record for Mary Augusta Andruss. The generated record for Mary, however, is incomplete, so *U* added the death date and place and also the funeral date as the burial date.

Extraction rules for the *Couple* and *Family* forms are generated similarly, except that the multiple entry fields in both forms call for an additional

type of generalization for lists. For the Abigail and Donald family in Figure 1, for example, when Mary Ely and Gerard Lathrop are added as children, GreenFIE not only generalizes each entry as explained above, it also generalizes the list itself to possibly have up to a user-specified maximum number of children (12 in our implementation). GreenFIE thus generates 12 extraction rules, the first with a capture group for the first child, the second with a non-capture group for the first child and a capture group for the second child, and so forth to the 12th in which the first 11 children are recognized in non-capture groups and the 12th is recognized with a capture group. When executed, GreenFIE takes the results and stitches all the children recognized into a single record with their parents. Lists are often numbered as they are in Figure 1, and GreenFIE knows about numbering schemes (e.g. Arabic Numerals, Roman Numerals, Ordinal Numbers) and replaces the left context, when it includes the numbering scheme as it does in Figure 1, with the correct number for each list item. There are no examples of persons with multiple spouses in Figure 1. On other pages in *The Ely Ancestry* (Vanderpoel 1902), when there are multiple spouses, they are numbered with ordinal numbers starting with the "2nd" spouse.

GreenFIE extraction rules are kept in a repository and executed in advance as a user moves from page to page. Eventually, the GreenFIE-generated extraction rules cover all but a very few exceptional cases. Thus, the extraction work of a user diminishes over time. The name "GreenFIE" stands for "**Green F**orm-based **I**nformation **E**xtraction, where "Green" is a designator for tools that improve themselves with use in real-world tasks (Nagy 2012).

### 3.4 GreenQQ: Extraction Rule Learning by Text Snippet Examples

Users interact with GreenQQ by means of sample snippets of the OCR'd text (Embley and Nagy 2017). For each lexical object set $S$ of a given ontology, users initially give GreenQQ a text-snippet example that contains a text instance $t$ to be extracted into $S$. The text snippet should also contain some surrounding text tokens that help identify and classify $t$ as being a member of $S$. From the text-snippet example, GreenQQ creates an extraction rule by generalizing and tagging the text tokens (e.g. identifying them as capitalized words, allcap words, $n$-digit numbers, punctuation) and designating the position within the snippet of the textual instance to be extracted. It then sweeps this extraction-rule template across an entire document whose text has previously been generalized and tagged and returns all matching instances for the specified lexical object set. Once bootstrapped in this way, GreenQQ is also able to discover and propose new extraction rules for other instances that likely should be extracted by the current set of extraction rules but were not. Presenting these proposed rules in terms of examples, the user can accept, reject, or modify these candidate rules. GreenQQ then executes the expanded rule set and proposes yet more potential rules. This cycle continues until the user is satisfied with the results.

As an example, consider applying GreenQQ to populate the *Person* ontology in Figure 7 from *The Ely Ancestry*, one of whose pages is in Figure 1. In preparation, GreenQQ tokenizes the full 830-page book and generalizes and tags each token according to its token type. Also, in preparation, a user considers the document and designates a few keyword tokens that are likely to help classify text instances, e.g. "b.", "born", "d.", "died", "m." for *The Ely Ancestry*. The user then gives an example text snippet for each lexical object set in the ontology for which information is available and designates the text instance to be extracted, e.g. [DeathDate "d. 1859." "1859"], which specifies that *DeathDate* is the target object set for the extracted text, "1859", within the text snippet example, "d. 1859.". From this example, GreenQQ generates the extraction template

```
DeathDate  d. NUM4 .  [1,1]
```

where `[1,1]` designates that the offset of the token(s) to be extracted within the template "`d. NUM4 .`" is 1 (zero start count) and that the number of tokens to extract is 1. When GreenQQ

sweeps this template pattern across the page in Figure 1, it extracts the death dates "1839", "1859", "1840", "1843", "1861", and another "1861", and it extracts hundreds more from the full book.

Continuing the example, GreenQQ next generates and classifies text snippet examples that it expects should be turned into extraction rules and presents a few of these for consideration. It identifies these candidate text snippets as frequent tagged text sequences surrounding user-given keywords and classified text tokens that *do not* include text tokens already labeled with the classification it is proposing. For example,

```
DeathDate: , b. 1812, d. 1882, son of
```

is one classified text snippet GreenQQ might return for consideration. From previous rules, GreenQQ has already associated the keyword `d.` with the `DeathDate` class, and the user-chosen window size of four tokens before and after the `d.` does not include a token sequence already labeled as a `DeathDate`. If the user now designates "1882" to be extracted with "d." and "," as the left and right context, GreenQQ generates the extraction rule

```
DeathDate  d. NUM4 ,  [1,1]
```

which when executed extracts "1882" and "1865" as additional *DeathDates* in Figure 1, and many more in the full book.

Figure 12 shows a set of GreenQQ-generated extraction rule templates that would extract most of the information of interest from Figure 1. Templates for Mrs. Lathrop's death date and place and her burial date (implied from the funeral date) are not included. Rule templates for this information can be created but are likely unique within the full book. Because Theodore's birth date was OCR'd as "i860" it is also not included in the information that would be extracted by the rule set. GreenQQ would see this OCR error frequently enough in the context of "b." that it might suggest

```
BirthDate: Theodore Andruss, b. i860. EOL 4 .
```

for consideration. In this case, the user could specify that `i860` should be extract with `b.` and `.` as its left and right context. If so GreenQQ would generate

```
BirthDate  b. ALPHANUM .  [1,1]
```

as a rule template.

Following the final rule create-and-execute cycle, GreenQQ groups the results into records from which it populates the ontology. GreenQQ only operates with ontologies that have a single non-lexical object set to which all lexical object sets are related, like the *Person*, *Couple*, and *Family* ontologies in Figures 7, 8, and 9. To form records, GreenFIE must group together all lexical objects related to a single non-lexical object and in the case when a connecting relationship set is $n$-ary ($n > 2$), as is the quaternary relationship set in Figure 8, must also properly group the connecting lexical objects. Since non-lexical objects are instantiated by ontological commitment, there are lexical objects in the text around which records are formed (e.g. person names for the ontologies in Figures 7 and 8 and a parent name for the ontology in Figure 9).

Depending on how book authors organize their presentation of information, grouping labeled text instances into records can be complex (Embley et al. 1999b). Sometimes record creation is straightforward, as it is for the Ely book in which the author groups all *Person*-ontology information immediately after the person's name. In this case, GreenQQ can run through its initial output, which is a lexical-object-set-name-labeled list of tokens in book-text order, and form record groups from one *Name*-labeled object to the next. At other times, however, records are intertwined. In Figure 1, for example, the couple Mary Ely and Gerard Lathrop is inside the text snippet that tells us that Abigail Huntington Lathrop and Donald McKenzie constitute a couple. In these cases, and in general, if we can process each record type in a separate run over the GreenQQ output, we can properly group labeled objects into records. In Figure 12 we have separated the templates into

```
Name           NUM5+ . CAP CAP CAP ,      [2,3]
Name           NUM5+ . CAP CAP CAP (      [2,3]
Name           m. NUM4 , CAP CAP ,        [3,2]
Name           NUM1or2 . CAP CAP ,        [2,2]
Name           m. NUM4 , CAP CAP CAP ,    [3,3]
BirthDate      b. NUM4                    [1,1]
DeathDate      d. NUM4                    [1,1]
GenderDes...   dau.                       [0,1]
GenderDes...   son                        [0,1]
```

(a)

```
Name           NUM5+ . CAP CAP CAP ,      [2,3]
Name           NUM5+ . CAP CAP CAP (      [2,3]
MarriageDate   m. NUM4                    [1,1]
SpouseName     m. NUM4 , CAP CAP . CAP (  [3,4]
SpouseName     m. NUM4 , CAP CAP ,        [3,2]
SpouseName     m. NUM4 , CAP CAP CAP ,    [3,3]
SpouseName     m. NUM4 , CAP CAP CAP .    [3,3]

Name           of CAP CAP CAP and         [1,3]
Name           of CAP CAP and             [1,2]
Name           of CAP CAP CAP CAP and     [1,4]
SpouseName     and CAP CAP ;              [1,2]
```

(b)

```
Parent1        NUM5+ . CAP CAP CAP ,      [2,3]
Parent1        NUM5+ . CAP CAP CAP (      [2,3]
Parent2        m. NUM4 , CAP CAP . CAP (  [3,4]
Parent2        m. NUM4 , CAP CAP ,        [3,2]
Parent2        m. NUM4 , CAP CAP CAP ,    [3,3]
Parent2        m. NUM4 , CAP CAP CAP .    [3,3]
Child          NUM1or2 . CAP CAP ,        [2,2]

Child          NUM5+ . CAP CAP CAP ,      [2,3]
Child          m. NUM4 , CAP CAP . CAP (  [3,4]
Child          m. NUM4 , CAP CAP ,        [3,2]
Child          m. NUM4 , CAP CAP CAP ,    [3,3]
Child          m. NUM4 , CAP CAP CAP .    [3,3]
Parent1        of CAP CAP CAP and         [1,3]
Parent1        of CAP CAP and             [1,2]
Parent1        of CAP CAP CAP CAP and     [1,4]
Parent2        and CAP CAP ;              [1,2]
```

(c)

*Figure 12: GreenQQ Templates for Information of Interest in Figure 1 for the (a) Person, (b) Couple and (c) Family Ontologies*

record groups and within each group have ordered the fields according to the Ely author's presentation. For the first *Family* record-template group in Figure 12(c), for example, GreenQQ would process its initial output file by finding a `Parent1` followed immediately by a `Parent2` and then immediately by a `Child` followed zero or more `Child`-labeled text instances without any intervening other-than-`Child`-labeled text. GreenQQ knows to look for more than one `Child` because of the 1-many relationship set from *Family* to *Child*. Grouping and ordering templates automatically is likely to be non-trivial in general and may require user input.

As with GreenFIE, "Green" is a designator for tools that improve themselves with use in real-world tasks (Nagy 2012). The "QQ" in "GreenQQ" stands for "Quick" and "Quality." The process of executing a rule set on an entire book and simultaneously proposing new rules for user consideration to be used in the next iteration is "Quick" enough to allow for real-time, synergistic user interaction. As we indicate in Section 8.1.2, if a book has reasonably well structured patterns, GreenQQ can "Quickly" generate "Quality" results.

## 3.5 ListReader: Extraction Rule Learning by Text Pattern Discovery

Like GreenQQ, ListReader (Packer 2014) processes a full book as a unit. It discovers record patterns in the text and generates extraction rules for them in a sequence of steps:

1. *Abstract Text*. ListReader replaces various sequences of one or more characters by a more abstract version of the character sequence. The string "4. Emma Goble, b. 1862.", for example, becomes

   `[Dg].[Sp][UpLo+][Sp][UpLo+],[Sp][Lo].[Sp][Dg][Dg][Dg][Dg].`

   Each digit becomes the symbol `[Dg]`, each word that begins with an uppercase letter becomes `[UpLo+]`, punctuation characters remain as themselves, and so forth.

```
[Dg].[Sp][UpLo+][Sp][UpLo+],[Sp][Lo].[Sp][Dg][Dg][Dg][Dg].
------------------------------------------------------------
...
2. Gerard Lathrop, b. 1838.
2. William Gerard, b. 1840.
4. Anna Margaretta, b. 1843.
5. Anna Catherine, b. 1845.
4. Emma Goble, b. 1862.
...
```

*Figure 13: ListReader Discovered Records*

2. *Align Text*. ListReader aligns text by finding identical sequences of abstract symbols. Figure 13 shows the text strings in Figure 1 that align with the abstract symbol sequence for Emma Goble. The ellipses at the top and bottom of the list stand for the hundreds of additional text strings in the book that also have the same sequence of abstract symbols.
3. *Identify Record Templates*. Not all text patterns ListReader discovers make good record templates. For our application a record pattern must contain either numbers or capitalized words or both. (Numbers and proper nouns, which in English are capitalized words, typically denote items of interest.) A record should contain at least two of these items of interest. (A record relates at least two items.) Record patterns should not contain long sequences of lower-case words. (Lower-case words in English list records tend to be delimiters, which are usually limited to just a couple of words.) Record delimiters such as newline characters, `\n`, are good indicators of record beginnings and endings. (All the text strings in Figure 13 have newline characters immediately preceding and immediately following each string.)
4. *Process Record Templates*. To turn record templates into extraction rules, ListReader must know how the patterns map to ontological conceptualizations. Following the principles of active learning (Settles 2012), ListReader selects the record group, which when labeled, will likely provide the most benefit. Typically large groups with lengthy strings that have good record characteristics are best. Because ListReader also does cross-record labeling for fields with field identifiers such as `b.` and `d.` for birth- and death-date fields in Figure 1, it also takes into account how much cross-record labeling can occur for a chosen record group. Once ListReader selects a record group it chooses a prototypical list element, finds the page it is on, and brings up the page in COMET on the right and the form for the ListReader ontology on the left. When a user then fills in the form record from the highlighted text, ListReader has the mapping it needs to generate a regular-expression ontology snippet extraction rule. For example, if ListReader highlights `4. Emma Goble, b. 1862` in Figure 1 and displays its ontology (Figure 3) as a form, the user should use COMET to extract `4` into the *ChildNr* field, `Emma` and `Goble` in the multiple-entry *GivenName* field, and `1862` into the *Year* field nested under *BirthDate*. ListReader can now label, without having to ask for the user's help, every `b.` field with just a year in every record template group in the same way.
5. *Generate Extraction Rules*. Given the information obtained by processing record templates, ListReader can now generate the extraction rule for the template. For the record template in Figure 13 ListReader generates

```
\n(\d{1})\.\s([A-Z][a-z]+)
\s([A-Z][a-z]+),
\sb\.\s(\d{4})\.
```

where the first capture group maps to *ChildNr*, the second and third map to *GivenName*, and the fourth maps to *Year* under *BirthDate*. Observe that `[A-Z][a-z]+` exactly mirrors `UpLo+`, that the digit-sequence lengths match, that the punctuation represents itself, and that the end-of-line record identifier, `\n`, has been added as the initial character signifying that these records always start on a new line.

As the name implies, ListReader discovers lists of records. It works well for semi-structured text like the text in Figure 1 and countless other family history documents, as well as many other types of semi-structured documents. It should not be applied to free-running narrative text.

### 3.6 GreenML/GreenDDA: Machine Learning of Extraction Rules

We are exploring the applicability of different approaches involving machine learning (ML) in recognizing and extracting named entities and family relationships from various text types. While annotators do exist for these data types, their off-the-shelf (OTS) performance derives from models trained on other types of annotated texts, particularly newswire articles. Performance on family history books suffers, especially for finding relationships. We believe we can improve on OTS performance in three ways.

First, GreenML is an ML approach that builds minimal models during training, based on high-quality annotations collected for a specific book via user interaction. The user annotates a page of a document using COMET as described earlier. Next, GreenML trains a model based on the results, which it then uses to annotate the next page. The user, through inspection and correction (where necessary) creates clean output, which is then added to the training set, triggering the creation of a new model for subsequent annotation. The cycle continues until the end of the book or until the user is satisfied that GreenML's accuracy is sufficient. The system is "Green" in the sense that it continuously improves its model as it receives user-checked and -corrected filled-in record forms page after page.

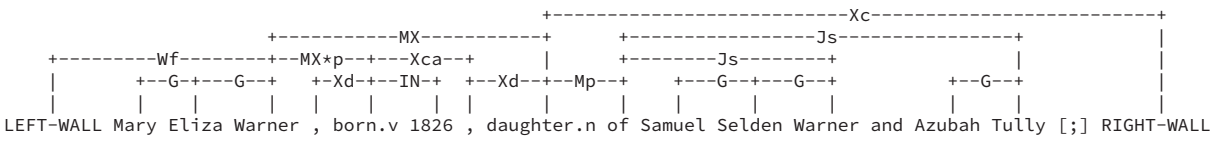
A second and related "Green" approach is called GreenDDA, for "**D**ecision **D**irected **A**daptation" (Nagy 2017). As with GreenML, a human supervises incremental, page-by-page training of models until some threshold point. In GreenDDA, though, the system proceeds from that point to annotate subsequent pages on its own, taking the results of each page and adding them (without human vetting) to the training set for model retraining. It then repeats this process on the remaining pages, extracting data from a full book.

Finally, instead of using an OTS ML model we could train our own model based on the totality of all COMET-user-verified data. This could be run without human intervention, or else with retraining via GreenML or GreenDDA as more data is collected.

Since most machine learning systems perform best with clean training data, we deem it advantageous to involve a human somewhere in the loop to check and correct generated data and to add missing data. Ideally, we could use active learning (Settles 2010) to target the most useful, informative interactions to present to the user for optimal annotation contribution, retraining models as needed to take advantage of this type of prioritization. If COMET is to be used to ensure that the data extracted from a book is complete and correct, some combination of green methods and active learning would behave like GreenFIE—one that would learn from a user's work and prepopulate subsequent pages with data in an attempt to reduce the user's workload.

### 3.7 OntoSoar: Extraction Rule Creation by Natural Language Processing and Cognitive Reasoning

OntoSoar (Lindes 2014; Lindes et al. 2015) extracts data using NLP techniques to discover and map extraction results from running text narrative. Its segmenter divides text into sentences or subsentential fragments that are then pipelined to the Link Grammar parser (Sleator and Temperley 1995). Figure 14 shows the parse of the fragment, "Mary Eliza Warner, b. 1826, dau. of Samuel Selden Warner and Azubah Tully;" from Figure 1.




```
Facts extracted:
  Reporting 7 objects:
    X2: Name(osmx85, "Mary Eliza Warner")
    X1: Daughter(osmx108)
    X1: Person(osmx108)
    X4: Name(osmx130, "Samuel Selden Warner")
    X3: Person(osmx137)
    X6: Name(osmx140, "Azubah Tully")
    X5: Person(osmx143)
    X7: BirthDate(osmx147, "1826")
  Reporting 6 relations:
    Y1(osmx150): Person(osmx108) has Name(osmx85)
    Y2(osmx153): Person(osmx137) has Name(osmx130)
    Y3(osmx156): Person(osmx143) has Name(osmx140)
    Y4(osmx159): Person(osmx108) was born on BirthDate(osmx147)
    Y7(osmx162): Daughter(osmx108) is a daughter of Person(osmx143)
    Y6(osmx165): Daughter(osmx108) is a daughter of Person(osmx137)
```

*Figure 14: OntoSoar Syntax Analysis (top) and Semantic Analysis (bottom)*

The Soar cognitive architecture (Laird 2012) analyzes the parse, extracting salient semantic objects and relations from the relationships represented by the parse links. The Soar engine in our implementation has 240 production rules. These rules build meaning using ideas inspired by construction grammars, which (1) pair textual forms with meaning; (2) construct knowledge structures with inference rules; and (3) map knowledge structures to ontologies by comparing their common entities and relationships. The mapping provides a conduit for populating the ontological conceptualization in Figure 2 with data. Figure 14 shows the results of semantically analyzing the Mary Eliza Warner text chunk.

## 4 Ontology Integration

Given the results of applying our ensemble of extraction tools (Section 3), our next task is to integrate the results into a single populated ontology. This requires both schema integration and data integration. For our genealogy application, the ontology in Figure 15 is our target for schema integration. The data that populates the target ontology should be integrated so that the same objects and relationships extracted into the source ontologies are represented only once the target ontology.

### 4.1 Schema Integration

General schema integration is known to be a hard problem—denoted by some as "AI-complete" or essentially unsolvable (Marie and Gal 2007). The major bottleneck is automatically discovering matching schema components (Noy 2004; Rahm and Bernstein 2001). Schema matching usually requires algorithms to be multifaceted, meaning that several schema integration techniques are used together, and machine-learned, because of the complexity of successfully weighing the evidence gathered from the multiple facets (Embley et al. 2001; Xu 2003).

Despite these general difficulties, the typical schema matching needed for an ensemble of extraction engines, each with their own ontologies, can be much simpler. Matching algorithms for ontology-based reading systems have the advantage of being able to observe the extraction of the same objects and the same relationships into the various object and relationship sets in the ensemble's collection of ontologies. "Same" here means that the text being extracted comes from the same location in the document, and thus with the assurance that it actually denotes the same object.

As an example, consider integrating the *Family* ontology in Figure 9 into the integrated target

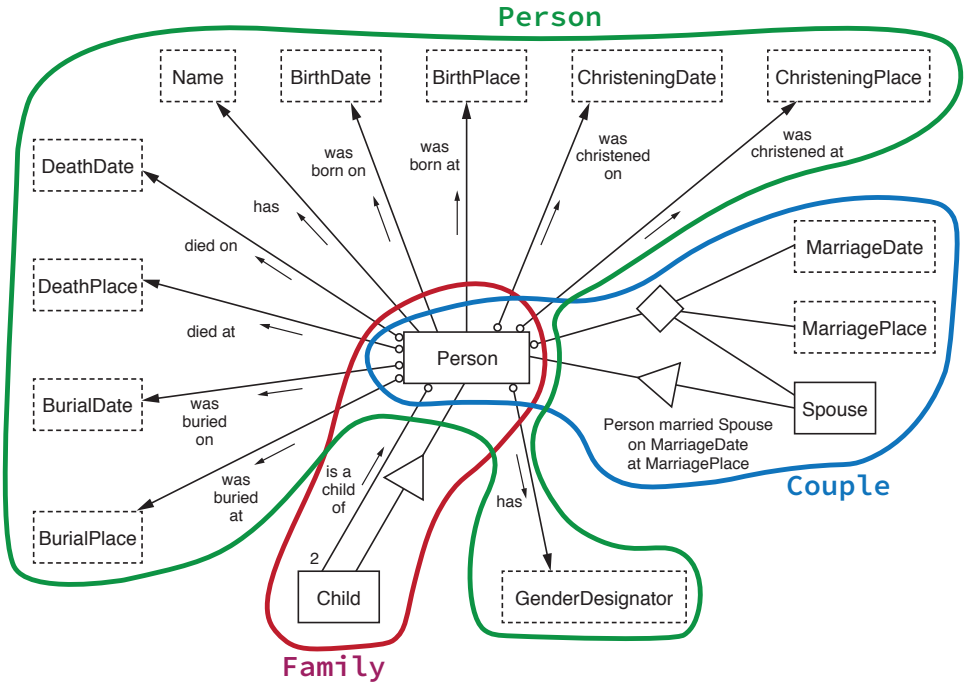



*Figure 15: Integrated Target Extraction Ontology*

extraction ontology in Figure 15. Note that except for *Child* none of the object set names match in the two ontologies. However, many of the same instances (e.g. "Charles Christopher Lathrop" and "Mary Augusta Andruss" in Figure 1) would be extracted into *Parent1* or *Parent2* in Figure 9 and into *Name* in Figure 2. Hence, *Parent1*, *Parent2*, and *Name* should all map to the same lexical object set in Figure 15, namely the *Name* object set. Similarly, *Child* should also map to the *Name* object set. Since *Name* is ontologically committed to *Person*, these mappings also indirectly create mappings to *Person* (and in the case of *Child* in Figure 9 also to *Child*).

Schema integration need not be fully automatic, but we must know how to map the tool extraction ontologies to the common integrated target extraction ontology. Schema mappings for the *Family* ontology are outlined above. For the *Couple* ontology *SpouseName* maps to *Name*, and indirectly to *Person* and *Spouse*. All other lexical object sets for our form ontologies (Figures 7, 8, and 9) map to lexical object sets with the same name. As the views superimposed on Figure 15 indicate, these three forms were designed to be complementary and to cover the integrated target extraction ontology. Except for the *Son* and *Daughter* components, the ontologies in Figures 2 and 5 map directly to the integrated extraction ontology in Figure 15. Sons and daughters are children, and in mapping them to *Child*, we also instantiate *GenderDesignator*s, "Son" and "Daughter" for them. For the ontology in Figure 3, we note that the integrated extraction ontology in Figure 15 does not have a breakdown of names and dates. Therefore, for each *BirthDate* and *DeathDate* object, we space-concatenate the strings in *Day*, *Month*, and *Year* as lexical *BirthDate* and *DeathDate* objects in Figure 15. Similarly, we space-concatenate *GivenName*s and the *Surname*. *ChildNr* is not in the integrated target extraction ontology, so we simply ignore it.

## 4.2 Data Integration

Lexical objects are the same if their text string is identical and is extracted from the same location in the document as determined by page and

page-offset index values. Non-lexical objects are instantiated by ontological commitment with lexical objects. Thus, non-lexical objects are the same when their declared lexical instantiators are the same. Since the integrated target extraction ontology in Figure 15 has *Person* and specializations of *Person* as its only non-lexical object sets, and since we instantiate Person objects by an ontological commitment with *Name*, we discover duplicate non-lexical objects by determining whether they have the same name. Thus, the fundamental data-integration problem we must resolve is to determine whether two *Person* names denote the same object.

We declare names to be the same if and only if their text strings are substantially the same and are located at the substantially the same place in the document. Names that refer to the same object but are not located at the same place in the document are resolved later. These resolutions either require coreference reasoning to determine, for example, that "Mrs. Lathrop" refers to Mary Augusta Andruss in Figure 1, or they require object identity resolution to determine, for example, that three of the four mentions of the name "Mary Ely" in Figure 1 refer to Gerard Lathrop's wife while the fourth refers to Gerard Lathrop's granddaughter.

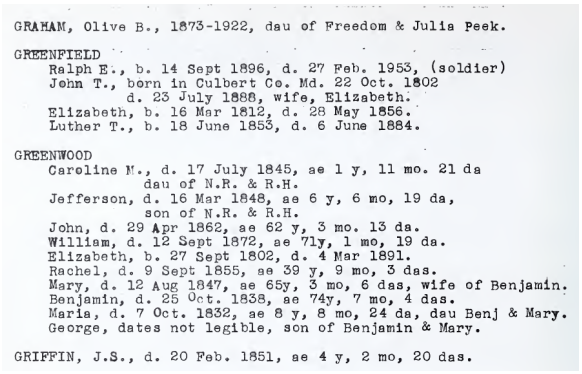
Although seemingly straightforward, determining "substantially the same" is often nontrivial. The extraction tools independently extract names; moreover, extraction rules within a single tool independently extract names. Often extracted names are identical, both offset and content, but there is no guarantee that a name in a document will be extracted in exactly the same way by all extraction rules within and across all tools.

Names can be extracted either as simple name strings or as complex name strings.

- Simple name strings comprise a single sequence of characters. Discrepancies can arise for several reasons: (a) One tool may extract name titles while another tool does not. In Figure 1, for example, one tool might extract "Judge Caleb Halstead Andruss" while another tool extracts "Caleb Halstead Andruss". (b) The tools are not consistent in the way they treat end-of-line hyphens. Some tools ignore them and thus would extract "Donald McKen" in Figure 1; some other tool may take the full name as given in the OCR text and extract "Donald McKen-\nzie"; and still some other tool may resolve the end-of-line hyphen and give the name as "Donald McKenzie". (c) Tools make mistakes and may extract "William Gerard Lathrop" in Figure 1, for example, as "William Gerard Lathrop, Boonton".
- Complex name strings comprise two or more name components that are not necessarily contiguous. Figure 16 shows some examples of names that do not consist of a single sequence of characters in the document: (a) "Freedom" "Peek" in the combined name "Freedom & Julia Peek"; (b) factored names such as "Ralph E." "GREENFIELD" and "John T." "GREENFIELD"; and (c) names such as "Benj" "GREENWOOD" and "Mary" "GREENWOOD", which are both factored and combined. Besides names, such as these, that are necessarily complex, tools may extract name strings that could be simple as complex name strings. Thus, for example, one tool may correctly extract the name that appears first in Figure 16 either as "GRAHAM, Olive B." or as "Olive" "B." "GRAHAM" or "Olive B." "GRAHAM". A tool may also incorrectly extract the name in several different ways, including as a partial name, "Olive" "GRAHAM", or an inferred name such as "Olive B." "Peek".

Determining whether two tool-extracted names are "substantially the same" is straightforward in the common case in which both the content and offset of all ordered name components match—and is otherwise anything but straightforward.

Heuristically, we determine whether two location-overlapping names match by first forming single-string interpretations of the names. These single-string interpretations include a resolution of end-of-line hyphens, both those included in the extracted text and those not included, but only if they immediately follow any one of the extracted


GRAHAM, Olive B., 1873-1922, dau of Freedom & Julia Peek.

GREENFIELD
Ralph E., b. 14 Sept 1896, d. 27 Feb. 1953, (soldier)
John T., born in Culbert Co. Md. 22 Oct. 1802
d. 23 July 1886, wife, Elizabeth.
Elizabeth, b. 16 Mar 1812, d. 28 May 1856.
Luther T., b. 18 June 1853, d. 6 June 1884.

GREENWOOD
Caroline M., d. 17 July 1845, ae 1 y, 11 mo. 21 da
dau of N.R. & R.H.
Jefferson, d. 16 Mar 1848, ae 6 y, 6 mo, 19 da,
son of N.R. & R.H.
John, d. 29 Apr 1862, ae 62 y, 3 mo. 13 da.
William, d. 12 Sept 1872, ae 71y, 1 mo, 19 da.
Elizabeth, b. 27 Sept 1802, d. 4 Mar 1891.
Rachel, d. 9 Sept 1855, ae 39 y, 9 mo, 3 das.
Mary, d. 12 Aug 1847, ae 65y, 3 mo, 6 das, wife of Benjamin.
Benjamin, d. 25 Oct. 1838, ae 74y, 7 mo, 4 das.
Maria, d. 7 Oct. 1832, ae 8 y, 8 mo, 24 da, dau Benj & Mary.
George, dates not legible, son of Benjamin & Mary.

GRIFFIN, J.S., d. 20 Feb. 1851, ae 4 y, 2 mo, 20 das.

*Figure 16: Part of a Page from Butler, County, Ohio, Cemetery Records (Stroup n.d.)*

name components in the original document's text. Then, by checking both content and offset of each token of this single-string interpretation of the name, if one of these interpreted names subsumes the other, we declare a match. If the subsuming name has a recognized name form (currently, either a last-name-first form or a standard form with a sequence of names preceded optionally with titles and followed optionally with suffixes such as "Sr." and "Jr."), it becomes the interpreted name for the merged *Person* object. Otherwise, the subsumed name becomes the interpreted name, unless it also fails to have a standard name form, in which case we take the nonstandard subsuming name as the interpreted name.

This process iterates over all location-overlapping names by comparing each name with the current best name, and eventually forms an equivalence class of *Person* objects to be merged as a single object with a single best interpreted name. Examples: (a) The interpreted name for all extractions of "Donald McKen-\nzie", whether they include the hyphen or not, become "Donald McKenzie". (b) "Judge Caleb Halstead Andruss" is chosen to be the interpreted name over the subsumed "Caleb Halstead Andruss". Similarly, "Olive B. GRAHAM" is chosen over the subsumed "Olive GRAHAM". (c) The names "Olive B. GRAHAM" and "Olive B. Peek" do not match even though they presumably refer to the same person, and their respective person objects are not merged. (d) "William Gerard Lathrop, Boonton" subsumes "William Gerard Lathrop" but is not a proper name form and is therefore rejected in favor of the subsumed name "William Gerard Lathrop".

Once an equivalence class of *Person* objects has been formed, we next look for duplicate relationships connected to this equivalence class. Checking for duplicate relationships involves determining whether the text objects in corresponding lexical object sets are substantially the same. We determine "substantially the same" for these text objects in the same way we determine whether two names are "substantially the same" except that instead of checking for acceptable name forms, we check for acceptable textual forms for dates and place names. Duplicate relationships are then discarded and replaced with a single relationship whose lexical objects are the best among the possibilities.

## 5 Ontological Constraints

The OSM conceptual modeling language is grounded in a decidable restriction of first-order logic (Embley and Zitzelberger 2010). This formal grounding enables us to specify and check constraints.

### 5.1 Ontology Language Formalization

We formally define OSM-OL (OSM Ontology Language) as a triple ($O$, $R$, $C$) where $O$ is a set of object sets, $R$ is a set of relationship sets, and $C$ is a set of constraints. Each object set in $O$ is a one-place predicate, and each object-set predicate has either a *lexical* or a *non-lexical* designation. Instances of lexical object sets are strings (e.g. *DeathDate(Nov. 4, 1898)*), and instances of non-lexical object sets are object identifiers (e.g. *Person(osmx17)* and *Spouse(osmx17)*). Each relationship set in $R$ is an $n$-place predicate ($n \geq 2$). We use a form of infix notation to specify instance relationships (e.g. *Person(osmx17) died on DeathDate(Nov. 4, 1898)*). $C$ is a set of constraints:

- *Referential*: object instances referenced in relationship instances must exist in referenced object sets, e.g.

$$\begin{aligned} &\textit{Person}(\textit{osmx17})\ \textit{died on} \\ &\quad \textit{DeathDate}(\textit{Nov. 4, 1898}) \Rightarrow \\ &\quad\quad \textit{Person}(\textit{osmx17}) \\ &\quad\quad \wedge\ \textit{DeathDate}(\textit{Nov. 4, 1898}) \end{aligned}$$

- *Participation*: instances in an object set $S$ must participate in relationships in a relationship set $R$ connected to $S$ according to declared participation constraints, e.g.

$$\begin{aligned} &\forall x(\textit{Child}(x)\ \Rightarrow \\ &\quad \exists^2 y(\textit{Child}(x)\ \textit{is a child of}\ \textit{Person}(y))) \end{aligned}$$

  specifies that every child has exactly two parents (hence the superscript 2).
- *Generalization/specialization*: instances in a specialization $S$ of a generalization $G$ must exist in $G$, e.g. $\forall x(\textit{Spouse}(x)\ \Rightarrow\ \textit{Person}(x))$.
- *General*: any predicate-calculus-specified constraint, e.g.

$$\begin{aligned} &\forall x \forall y \forall z( \\ &\quad \textit{Person}(x)\ \textit{was born on}\ \textit{BirthDate}(y) \\ &\quad \wedge\ \textit{Person}(x)\ \textit{died on}\ \textit{DeathDate}(z) \\ &\quad\quad )\Rightarrow\ y \le z \end{aligned}$$

  specifies that a person's death date must not precede the person's birth date.

Note that all but general constraints can be specified in OSM's graphical notation. Nevertheless, all constraints are just predicate calculus statements. Note also that the graphical black-triangle aggregation symbol has no associated constraint because aggregation is merely a visual grouping of relationship sets devoid of other formal meaning.

The constraints mentioned so far are all crisp, yielding only a strict "yes" or "no" to determine violations. We call these constraints *hard*. Ontologies that seek to model reality, however, should also provide for *soft* constraints. By introducing probability distributions, we can allow constraints to return the probability that an assertion holds, and thus have soft as well as hard constraints. For participation constraints, instead of a crisp *min*:*max* designation, we can return the probability of the actual cardinality as asserted. Thus, for example, based on the probability distribution, we can determine how reasonable it is that a mother has 2 children (or 17 or 209). With extended general constraints, we can check the sensibility of many assertions such as whether the age of a mother is reasonable for having a child or whether the age difference between spouses is common for the time and place of their marriage.

Referential-integrity constraints and is-a constraints in generalization/specialization hierarchies should not be extended to allow for uncertainty. The model itself would not make sense if objects referenced in relationships do not exist or if objects in specializations are not also in their generalizations (e.g. if an individual is a *Child* or a *Spouse* but not also a *Person*).

## 5.2 Constraint Checking

Given an ontological model of our world of interest, we can check extracted assertions against this model to see if they make sense. Unlike standard databases, which only allow updates if no constraints are violated, we allow our extraction tools to populate an ontology independent of whether they violate hard constraints or whether they are unreasonable with respect to soft constraints. We then determine whether the extracted assertions make sense with respect to the constraints of the ontological world of interest (Woodfield et al. 2016).

Figure 17 shows a soft constraint written in the OSM-OL ontology language of the integrated target extraction ontology in Figure 15. It returns the probability of a child having been born to a mother at a particular age. The first three antecedent statements are predicates that come directly from the ontology in Figure 15. The fourth implicitly adds an object set *Gender* and a relationship set *Person has Gender* to the ontology. We populate the implicit object and relationship

set with instances determined from information at hand: (1) If *Person*($x$) has a *GenderDesignator*, we immediately know the *Gender*. (2) If not, but the *Person* is married and the spouse's gender is known, the person's *Gender* is known. (3) If still unknown, the first given name maps to the probability of the person being male or female in a large frequency table created by running over the billion or so name/gender pairs in *Family Tree* (FamilySearch n.d.). If the probability is above a specified threshold (currently set at 0.95), we can confidently set the gender. (4) Finally, if still unknown, we leave the gender unknown. The fifth antecedent statement makes use of the data-frame declared internal representation and operators for dates. The final antecedent statement references a probability distribution, which we are able to compute over the many millions of mother-child relationships in *Family Tree*. The consequent statement yields a relationship for an implicit quaternary relationship set that connects the object sets *Person* and *Child*, which are already in the ontology, and *Age* and *Probability*, which implicitly belong to the ontology.

*Child*($c$) *is a child of Person*($m$)
$\wedge$ *Person*($c$) *was born on BirthDate*($d_1$)
$\wedge$ *Person*($m$) *was born on BirthDate*($d_2$)
$\wedge$ *Person*($m$) *has Gender*(*Female*)
$\wedge$ *Age*($a$) = *Age*(*YearOf*($d_2$) − *YearOf*($d_1$))
$\wedge$ *mother's Age*($a$) *at child's birth has Probability*($p$)
$\Rightarrow$
*Person*($m$)*'s Age*($a$)
*at Child*($c$)*'s birth has Probability*($p$)

*Figure 17: Probability of a Mother's Age at Her Child's Birth being Reasonable*

## 5.3 Constraint Violation Resolution

The process of detecting and correcting or removing inaccurate information in a data repository is known as data cleaning (Müller and Freytag 2003; Rahm and Do 2000). For our reading system, data cleaning consists of observing constraint violations and resolving them. Given the results of constraint checking, the reading system can sometimes correct itself. Most often, however, constraint violation resolution requires human intervention.

Figure 18 shows the interface a human uses to resolve constraint violations. Hovering over a record highlights its fields and the information in the text document filled into the various record fields. It also marks fields identified as possibly being in error with a red, yellow, or green warning icon, depending on the severity of the constraint violation—red when something is definitely wrong such as a death date preceding a birth date, yellow when something is likely wrong, and green when something is likely right but should be double-checked, such as when an OCR error (e.g. "i860") has been automatically corrected. Often, the highlighting is sufficient to indicate the error. In Figure 18 a user can easily see that Mary and Gerard are not children of Joel and Mary Gloyd and should click on the corresponding red-x button to remove the highlighted record.

If a user clicks on a warning icon, an explanation box pops up. In Figure 18, the user has clicked on the warning icon in the field filled in with "Mary Ely". Three messages apply: (1) Mary Ely has too many parents, (2) Mary was born 14 years before her presumed parents Mary Eliza Warner and Joel M. Gloyd were married, and (3) Mary Ely's presumed mother Mary Eliza Warner would have been only 10 years old when Mary Ely was born.

All our soft general constraints are implication statements. When implication statements are violated, one or more of the antecedent statements must be incorrect. The suspect antecedent statements are those corresponding to assertions stored in the ontology. Explanations, such as those in Figure 18, list these statements as possibly being the cause of a constraint violation. Note that we can generate these human-readable explanation statements from the rule itself by replacing references to non-lexical objects by their ontologically committed lexical counterparts (person names in our application) and by replacing references to lexical objects by placing the lexical values in parentheses following the object set name. In

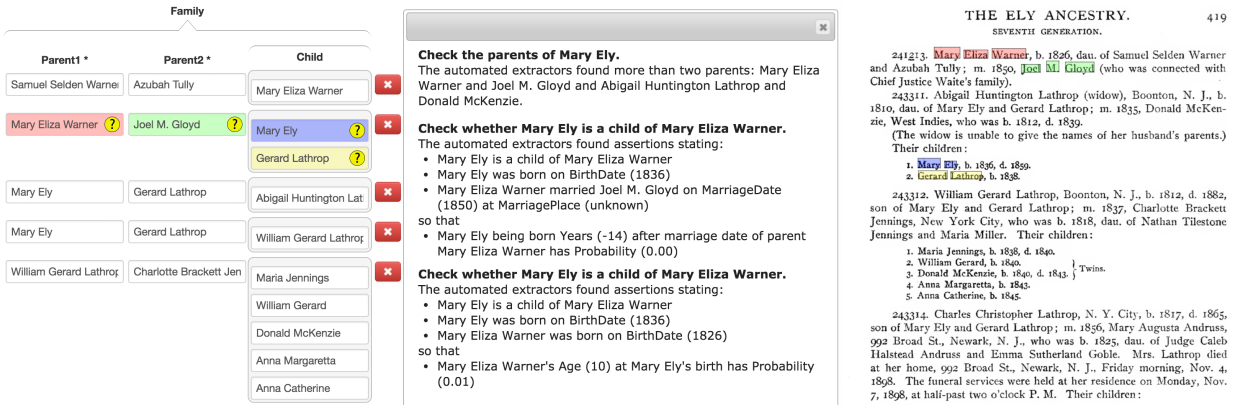



*Figure 18: Error Warning Indicators and Explanations*

Figure 18, the asserted antecedent statement declaring that Mary Ely is a child of Mary Eliza Warner is wrong.

Automatic retraction of assertions is sometimes possible. When an implication rule has only one antecedent statement corresponding to an assertion stored in the ontology, the assertion can safely be retracted. Although not quite as safe, a single antecedent assertion in the intersection of multiple rule violations is almost certain to be wrong and can be retracted. In Figure 18, for example, the antecedent assertion *Person(Mary Ely) is a child of Person(Mary Eliza Warner)* is in both the born-much-earlier-than-marriage violation and the mother-too-young-to-give-birth violation and should be retracted. Sometimes, we can heuristically determine which assertion(s) should be retracted. We have observed, for example, that when a child has too many parents and all the parents precede the child in the document flow, the closest couple or single parent is correct, and parent-child assertions for all other parents can be retracted. In the document in Figure 18 the four mentioned parents all precede Mary Ely, and her correct parents, Donald and Abigail McKenzie, are closer to Mary than her incorrect parents, Joel and Mary Gloyd.

When a human is in the loop to check and correct extracted assertions, the system automatically retracts identifiably incorrect assertions before presenting results for an initial check-and-correct session. Warning icons are also initially omitted, which lets users do unbiased checking and correcting and avoids overwhelming them with largely obvious and often extraneous statements about what might be wrong. However, if constraint violations remain after the initial human check-and-correct session, no retraction takes place, icons are added as a warning that something has likely been overlooked, and the document is returned to the user for further checking.

## 6 Ontological Inference

Authors of factual documents often convey information by implication and expect readers to infer these facts by what is explicitly stated. In Figure 1, for example, there are several implications of interest about person names. Maria Jennings' maiden name is Maria Jennings Lathrop, the surname being added by implication based on cultural norms. Abigail Huntington Lathrop would have been known in her married life as Abigail McKenzie since she is female (not stated, but determined by implication) and was married to Donald McKenzie. The author does not explicitly sort out the identity of the persons named "Mary Ely" in Figure 1, but leaves it to the reader to determine that there is one person named Mary Ely who is married to Donald Lathrop and another who is her granddaughter.

Reading systems, like human readers, need to be able to "read between the lines" and infer

implied information (Embley et al. 2016). Reading systems should be able to infer new objects and relationships, placing them in potentially new ontological object and relationship sets. They should also be able to resolve object identity and determine which non-lexical object identifiers denote the same object.

### 6.1 Infer New Objects and Relationships

Because OSM extraction ontologies are formally grounded in first-order logic, it is natural to infer new objects and relationships as Datalog-like queries (Datalog User Manual 2004; Gallaire and Minker 1978). Although we have used Datalog directly to derive information (Embley et al. 2016; Park 2015), we currently program the equivalent of Datalog queries to infer the specific implied information we seek.

Figure 19 shows our target extraction ontology extended with four new object sets: *InferredGender*, *InferredBirthName*, *InferredMarriedName*, and *InferredFormalName*, which is an aggregate of one or more *GivenName*s and *Surname*s, and zero or more *Title*s and *Suffix*es. The reading system populates these new object sets and their connected relationship sets by inference.

In our application we first infer inverses of persons and their spouses. If *Person(x) married Spouse(y) on MarriageDate(z) at MarriagePlace(w)*, then *Person(y) married Spouse(x) on MarriageDate(z) at MarriagePlace(w)*. No new object or relationship sets are created but new facts not directly stated are added. From Figure 1 the extraction engines would have read that Mary Eliza Warner married Joel M. Gloyd, which implies also that Joel M. Gloyd married Mary Eliza Warner.

Next we obtain *InferredGender* as explained earlier. Extracted *GenderDesignator*s are converted into their appropriate gender values. For persons without *GenderDesignator*s, the system infers gender by first given names when the certainty is sufficiently high, relying also on spouse gender for married persons. Knowing the gender is particularly important for inferring names.

Before populating inferred name object sets, we first standardize all names. At this point in our processing pipeline, we have both the original text (as extracted) and the interpreted text (a single string of space-separated components in which components with end-of-line hyphens have been closed up). Standardized text is a third representation of all lexical objects. For names, we standardize each name component with upper- and lower-case letters and order the components respectively by titles (if any), given names, surname, and suffixes (if any). Thus, for example, the name "ALBRIGHT, ESTHER R." in Figure 20 becomes "Esther R. Albright" in its standardized form.

With names in standard form, we can determine the surname of fathers and husbands and, using familial relationships, reason about birth names and married names. From the information in Figure 20, for example, we can determine that "Esther R. Albright" is a married name since she is female and married to Winfield S. Albright. We can also determine that Esther's maiden name is Esther R. Morris since her father is Thomas Benton Morris. An *InferredFormalName* is a person's birth name with titles and suffixes (if any) attached and with the surname extended with additional married surnames (if any). Thus, Esther's *InferredFormalName* is "Esther R. Morris Albright" where both "Morris" and "Albright" are surnames.

### 6.2 Infer Object Identity

One way to infer object identity is by coreference resolution. Often several linguistic expressions (e.g. noun phrases, proper nouns and pronouns) in a text may refer to the same entity under discussion. In Figure 1, the expression "Mrs. Lathrop" corefers to the person previously mentioned as "Mary Augusta Andruss." Finding and resolving instances of coreference is a difficult NLP problem. A wide range of knowledge sources, usually in combination, serve in systems that process text and posit coreference: morpholexical features such as person and number agreement; syntactic configurational relationships like apposition and pronoun

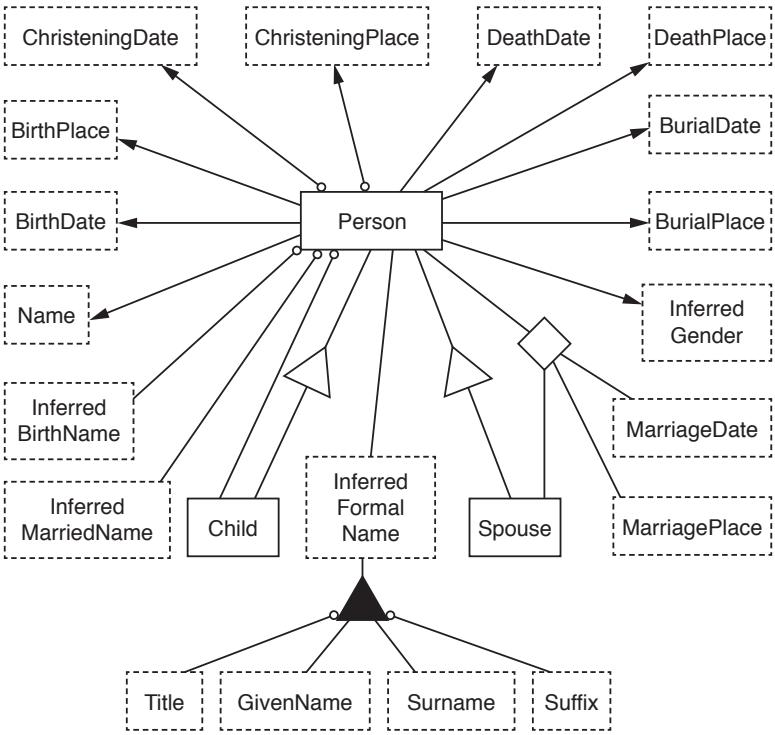



*Figure 19: Ontology with Inferred Object and Relationship Sets*

ALBRIGHT, ESTHER R. d 1 Jan 1946 113 Sherman St Dayton OH BD Abbottsville Cem Dke Co OH 3 Jan 1946 b 22 July 1863 Butler Co OH age 82-6-9 f THOMAS BENTON MORRIS Butler Co OH m ANGELINE HARROD Hamilton Co OH housekeeper widow sp WINFIELD S. ALBRIGHT 1 daughter Mrs HENRY RANCH 4 sons HENDERSON of Greenville WILBUR of Greenville GEO of Dayton ELBERT of Dayton 12 grandchildren 2 brothers ARTHUR MORRIS Venice OH & SAM MORRIS Harrison OH 2 sisters Miss ELLA MORRIS Greenville OH & Mrs ADA HARP Tulsa OK

*Figure 20: Esther Albright Funeral Home Record (Miller Funeral Home Records, 1917 – 1950, Greenville, Ohio 1990)*

binding; semantic properties like scope, modals and negation; pragmatic properties like animacy and gender; text-related properties like register and stylistics; and memory-related functions like recency, distance, and repetition. Rule-based implementations have existed for some time now (e.g. (Grosz et al. 1995)), but more current work also focuses on statistical, machine learning (Recasens and Hovy 2009), and deep learning solutions (Clark and Manning 2016).

Another way to infer object identity is by matching object properties and relationships with other objects (Benjelloun et al. 2009; Bhattacharya and Getoor 2007; Elmagarmid et al. 2007). In Figure 1, for example, our reading system will have extracted four different persons with the name "Mary Ely" and will have also extracted several related items of information for each of them. Three of the Mary Ely's have a spouse named Gerard Lathrop, albeit each with a different child: Abigail Huntington Lathrop, William Gerard Lathrop, and Charles Christopher Lathrop. The fourth Mary Ely has a birth year, 1836, a death year, 1859, a father, Donald McKenzie, and a mother, Abigail Huntington Lathrop (who by earlier data integration is known to be the same person as the first Mary Ely's daughter). This information is sufficient for Duke (Duke: Fast Deduplication Engine n.d.), an off-the-shelf entity-resolution engine, to conclude that the three Mary Ely's married to a Gerard Lathrop are all the same person and are not the same person as the Mary Ely who is the daughter of Abigail Huntington Lathrop.

In our application, we have yet one more way we can resolve object identity. Sometimes the persons for whom we are extracting information are already in *Family Tree*. In this case we can find potential matches in the tree, gather related information from both *Family Tree* and the document being read, and present it for consideration for resolving object identity. Figure 21 shows an example in a D-Dupe-like view (Kang et al. 2008) of the information regarding Mary Ely. In the *Person* object set on the left are the four Mary Ely's under consideration for merging along with their attribute values (if any). In the *Person* object set on the right are two possible matches in *Family Tree*. One-hop relationships for all these persons are are also found and displayed. When related persons are also found to be possible duplicates, they are grouped together. Those that have matches between the document being read and *Family Tree* appear in the middle. The evidence in Figure 21 is even more persuasive for the merge of the first three Mary Ely's than the evidence in the document alone. Moreover, the evidence also argues for a merge within *Family Tree* of Mary Ely (KFRL-WXZ) and Mary Eli (MGV1-9BJ).

## 7 Ontology Construction

Reading to construct ontologies is perhaps the most complex aspect of automated reading systems (Cimiano 2006; Cimiano et al. 2006; Wong et al. 2012). Nevertheless, for some special cases, we can augment or integrate ontologies in our collection, and with some restrictive types of input documents, we can sometimes construct ontologies from the text itself.

### 7.1 Connect and Augment Existing Conceptualizations

Named entity recognition (NER) systems, for example the Stanford CoreNLP machine learning annotator (Finkel et al. 2005), identify references to entities in text. They flag and categorize proper noun expressions as referring to such objects as persons, locations, organizations, and time expressions (Nadeau and Sekine 2007). Using the principle of ontological commitment, tagged entities from NER output can populate an ontology snippet consisting of a non-lexical object set named by the entity type connected to a lexical object set giving the entity's designating name. Figure 22(a) shows an example for *Location*.

Running over the text in Figure 1, a *Location* entity recognizer would populate the ontology in Figure 22(a) with entities for "West Indies", "Boonton N. J.", "N. Y. City", "Newark, N. J.", "New Jersey", and "Elizabethtown". Now, notice that an NLP system would be able to discover that the first three of these locations are related

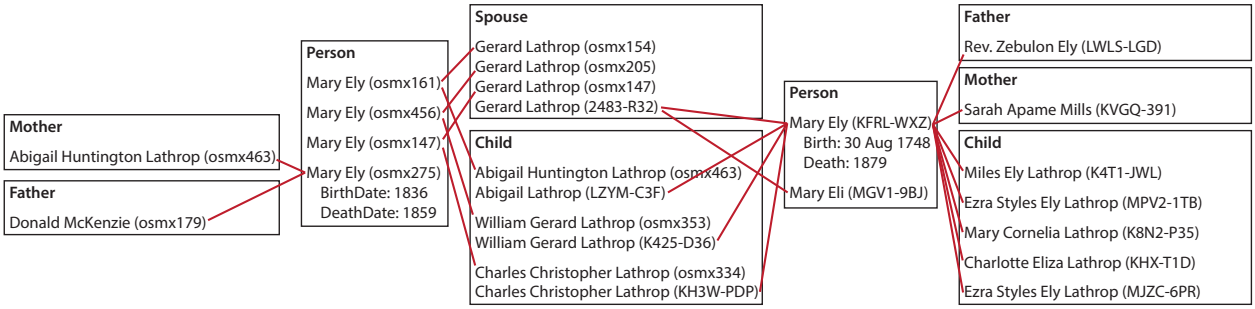



*Figure 21: Duplicate Detection and Resolution*

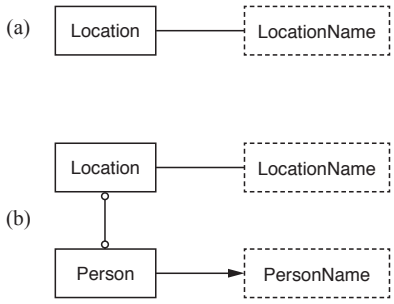



*Figure 22: Location Ontology*

respectively to Donald McKenzie, William Gerard Lathrop, and Charles Christopher Lathrop. Hence, we can create and populate a relationship set connecting *Location* in Figure 22(a) and *Person*, the primary object set in the ontologies we have been discussing. Not knowing what the relationship is, we set the constraints to be as loose as possible: many-many and optional on both connections. Figure 22(b) shows the result.

Similarly, we can connect the *Organization* ontology snippet in Figure 23(a) with the *Person* object set based on the sentences linking Emma Goble to organizations in the paragraph about her in Figure 1. We may even be able to do better: (1) we may already have a more extensive *Organization* ontology with object sets *Member*, *MemberName*, and *Officer* as a specialization of *Member* and instantiated by identifying the *Member* and *Office* in the *Organization*; or (2) perhaps by a more complex NLP analysis of the sentences in the paragraph about Emma Goble, we could construct such an ontology. Then, after populating both our *Person* ontology and the *Organization* ontology and by coreference resolution discovering that all the references to Emma Goble refer to the same person, we can integrate the two ontologies as Figure 23(b) shows.

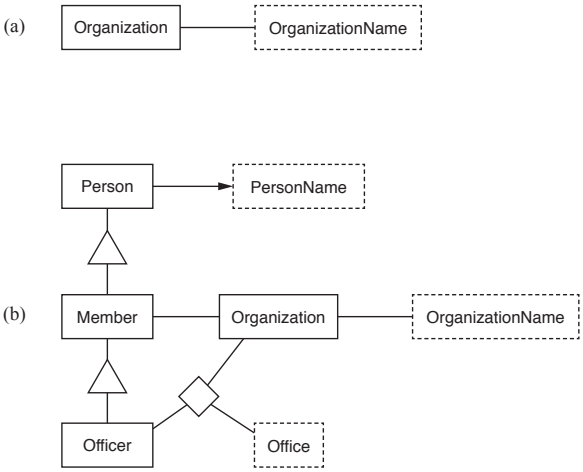



*Figure 23: Organization Ontology*

## 7.2 Read Semi-structured Text to Create Conceptualizations

TANGO (Table ANalysis for Generating Ontologies) is a methodology we have proposed as a means of automatically deriving an ontology from a correlated collection of standard tables (Tijerino et al. 2005). In essence, the idea is to find a correlated collection of ordinary tables, reverse-engineer them into conceptual models, and integrate them into a single conceptual model—an ontology that represents their union. The tables in a relational database, which are by definition

correlated and furthermore have often been created from a conceptual model, are particularly amenable to TANGO processing.

As an example, we illustrate how we can use the TANGO methodology to derive an ontology starting from a document, which we assume is semi-structured like the document in Figure 1. As Figure 13 shows, ListReader (Packer 2014) can find text patterns in semi-structured text that have record-like structure. When using ListReader for information extraction, a user maps a selected text record to a given form to establish the connection between the components of the text record and the fields of a form, which thus also establishes the mapping of a text record to an ontology. When reading to create an ontology, however, there is no form and no ontology. In this case, a user can turn a ListReader-discovered list of text records into a relational table by associating user-chosen attribute names with ListReader-generated abstract text components. In Figure 13, for example, a user can associate *Name* with `[UpLo+][Sp][UpLo+]` and *BirthDate* with `[Dg][Dg][Dg][Dg]`. Stripping away the text in the records that does not associate with any of the attributes yields a relational database table. To name the objects the table represents, the user provides a name for the table—*Person* for our example. Figure 24(a) shows the result for Figure 13.

Reverse engineering the relational table in Figure 24(a) yields the snippet of the ontology in Figure 7 consisting of just the object sets *Person*, *Name*, and *BirthDate*. Other ListReader-discovered text record patterns in Figure 1 and elsewhere in *The Ely Ancestry* would lead to the relational tables *Person*(*Name, GenderDesignator*) and *Person*(*Name, DeathDate*). These relational tables along with other *Person*-property tables that include christenings, burials, and place information for births and deaths integrate trivially and can be reverse-engineered into the ontology in Figure 7. Obtaining the finer properties of the ontologies such as participation constraints can only be done heuristically. In practice, a knowledgeable user should check and, as necessary, adjust these finer properties of generated ontologies.

Although beyond the capabilities of ListReader's current implementation, an extension of ListReader's pattern discovery mechanism could potentially generate nested database relations like those in Figures 24(b) and 24(c). (Currently hand-written or GreenFIE-generated regular-expression rules can extract relations with these nested patterns.) Reverse engineering the nested relations in Figures 24(b) and 24(c) respectively yields the ontologies in Figure 8 and Figure 9. Integrating these ontologies and the ontology in Figure 7 yields the ontology in Figure 15.

### 7.3 Construct Conceptualizations from Text Specifications

Generating a natural language description of an OSM ontology is straightforward. We simply write down each relationship-set name and each generalization/specialization *is-a* connection in some arbitrary order. Figure 25(a) shows a sampling of sentences describing the ontology in Figure 2. Other researchers have also studied verbalization techniques for conceptual models with the goal of making it easier, for example, to validate models with domain experts. There are interesting challenges in creating high-quality verbalizations, but the approaches are generally relatively straightforward (Curland and Halpin 2012; Halpin 2004; Halpin and Curland 2006). Reversing the process, however, to generate an ontology from ordinary prose is non-trivial.

Although not quite ordinary natural language prose, we have developed a model-equivalent specification language for OSM conceptual models (Embley 1998; Liddle et al. 1995). Model-equivalent languages provide a way to specify ontologies using natural-language-like statements, but are written according to a restricted context-sensitive grammar. Figure 25(b) gives an example. Object set names are nouns and are capitalized. Verb phrases, which are written in lower-case words, connect nouns to form sentences. Sentences define relationship sets. Constraints appear in square brackets. Participation constraints for object-set/relationship-set connections appear in sentences in square brackets as Figure 25 shows.

| Person(Name, | BirthDate) |
|---|---|
| ... | |
| Gerard Lathrop | 1838 |
| William Gerard | 1840 |
| Anna Margaretta | 1843 |
| Anna Catherine | 1845 |
| Emma Goble | 1862 |
| ... | |

(a) *Relational Table Obtained from the ListReader-Captured Data in Figure 13*

| Couple(Person, | (SpouseName, | MarriageDate, | MarriagePlace)) |
|---|---|---|---|
| Thomas Patterson | Jane Clark | | |
| Ben Ezra Stiles Ely | Elizabeth Eudora McElroy | 1848 | |
| | Abbie Amelia Moore | 1873 | |
| Harriet Clarissima Ely | Beale Steenberger Blackford | | |
| Zebulon DeForest Ely | Clara Vanola Major | 1874 | |
| | Mamie Anna Souder | 1878 | |

(b) *Nested Relational Table of Couple Information from Page 421 of The Ely Ancestry*

| Family(Parent1, | Parent2, | (Child )) |
|---|---|---|
| Samuel Selden Warner | Azubah Tully | Mary Eliza Warner |
| Mary Ely | Gerard Lathrop | Abigail Huntington Lathrop |
| Abigail Huntington Lathrop | Donald McKenzie | Mary Ely GerardLathrop |
| Mary Ely | Gerard Lathrop | William Gerard Lathrop |
| William Gerard Lathrop | Charlotte Brackett Jennings | Maria Jennings |
| | | William Gerard |
| | | Donald McKenzie |
| | | Anna Margaretta |
| | | Anna Catherine |
| Nathan Tilestone Jennings | Maria Miller | Charlotte Bracket Jennings |
| ... | | |

(c) *Nested Relational Table of Family Information from Figure 1*

*Figure 24: Relational Database Tables Created from Text Patterns Found on Page 419 and 421 of The Ely Ancestry*

```
Person has Name.
Person was born on BirthDate.
Person was born at BirthPlace.
Child is-a Person.
Child is a child of Person.
Spouse is-a Person.
Person married Spouse on MarriageDate at MarriagePlace.
...
```

(a) *Natural Language Rendition of the Ontology in Figure 2 (partial).*

```
Person[1] has Name[1:*].
Person[1] was born on BirthDate[1:*].
Person[1] was born at BirthPlace[1:*].
Child is-a Person.
Child[2] is a child of Person[0:*].
Spouse is-a Person.
Person[0:*] married Spouse[1:*] on MarriageDate[1:*] at MarriagePlace[1:*].
...
```

(b) *Textual Specification of the Ontology in Figure 2 (partial).*

*Figure 25: Ontologies Rendered and Specified in Natural Language*

Generalization/specialization hierarchies are written as "is-a" sentences with the specialization as the subject and the generalization as the object in the sentence.

Many researchers have studied the challenge of moving from textual descriptions to corresponding conceptual models or ontologies. For example, Chen studied a number of the foundational issues associated with moving from English natural language specifications to ER diagrams (Chen 1983). Cimiano et al. have proposed numerous techniques for constructing ontologies from text (Buitelaar et al. 2008; Cimiano 2006; Cimiano and Völker 2005; Cimiano et al. 2006). Notably, Heinrich Mayr and his colleagues have pursued a long line of research into how to better correlate natural language statements of requirements specifications to conceptual models that can support information systems development (Fliedl et al. 2003, 2005, 2007, 2004; Kop et al. 2004; Mayr and Kop 1998). KCPM, the Klagenfurt Conceptual Pre-design Model, supports a "conceptual pre-design" step in the software development lifecycle in order to bridge the historical "impedance mismatch" between requirements analysis and conceptual design. Mayr and colleagues provide semi-automatic techniques for mapping from natural language requirements specifications to pre-design schemas, and from pre-design schemas to conceptual schemas and domain ontologies.

We anticipate that researchers will continue to work on this interesting and complex challenge for decades to come.

## 8 Project Status

In the domain of family history and in cooperation with FamilySearch International (FamilySearch n.d.), we have implemented a genealogical document reading system. We give here the implementation status of this system including the status of the extraction tools and of the pipeline that integrates the information received from the extraction engines, checks it with respect to ontological constraints, standardizes it, infers new additional information not directly extractable from the document, and generates artifacts ready for import into *Family Tree*. We also discuss our initial experience with importing the information

obtained by the reading system into *Family Tree*. Figure 26 shows the management interface to our project, from which an administrator can control the reading system. The menu on the left allows the administrator to import a new book to be read, configure and test extraction tools, and manage the process of reading the book and generating GedcomX files (Gedcom X n.d.) for import into *Family Tree*.

Besides our near-term objectives within FamilySearch, we also envision applying our reading system to construct, populate, and query a *Web of Knowledge* (*WoK*) for any domain of interest and thus also for a collection of overlapping and non-overlapping domains of interest. We report briefly on the status of this effort and particularly on the use of reading systems to "understand" queries and map them to formal queries over a WoK.

## 8.1 Extraction Tools

Figure 27 shows the extraction tools within the scope of our project categorized by methodology type and ordered by the amount of human effort required to configure the tools to read a document. Wanting to cover the space of document types from those that are highly structured (e.g. the document in Figure 28) to those with free-running unstructured text (e.g. Figure 29) to everything in between (e.g. Figure 1) prompted us to develop tools in a variety of methodology paradigms. Within these paradigms our research efforts are directed toward reducing human involvement—generating expert system rules from examples and discovered text patterns, learning how to map parsed sentences to ontologies, and reducing the amount of training data needed for machine learning.

In Figure 27, tools in red (FROntIER, OntoES, GreenFIE, ListReader, OntoSoar) have been developed and evaluated as academic prototypes. Below, we give evaluation results of these tools and assess their strengths and weaknesses. Tools in green (GreenQQ, GreenML, GreenDDA, OntoSoar2) are academic prototypes in development. Below, we briefly mention our expectations for these tools. In violet are language machine learning paradigms. We may consider adopting or adapting some of the tools developed in these areas for our reading system. We have yet to do a tech-transfer of any of these tools to our administrative management system (Figure 26).

### 8.1.1 Completed Academic Work

*FROntIER*

Nearly 20 years ago we began building ontology-based extraction systems (Embley et al. 1999a, 1998a,b). These systems extracted named entities from small record-like write-ups like car ads and obituaries. The extracted entities from these write-ups were assumed to all be related to the primary object, e.g. the car being advertised for car ads and the deceased person for obituaries. As explained in Sections 3.1 and 2.2, FROntIER extends this work to extract and populate relationships and ontology snippets (Park 2015).

In an evaluation of FROntIER extraction, we developed entity and relationship extraction rules sufficient to correctly extract the information from *The Ely Ancestry* Page 419 in Figure 1 with respect to the ontology in Figure 2. We then applied the extraction ontology to Page 479 and obtained the results in Table 1.

*Table 1: FROntIER Results*

| | Prec. | Rec. | F-score |
|---|---|---|---|
| Name | 1.00 | 1.00 | 1.00 |
| BirthDate | 1.00 | 0.96 | 0.98 |
| DeathDate | 1.00 | 1.00 | 1.00 |
| MarriageDate | 1.00 | 1.00 | 1.00 |
| born on | 0.92 | 0.82 | 0.87 |
| died on | 0.75 | 0.75 | 0.75 |
| son of | 1.00 | 0.83 | 0.91 |
| daughter of | 0.67 | 0.33 | 0.44 |
| child of | 0.79 | 0.59 | 0.68 |
| married | 1.00 | 0.50 | 0.67 |

FROntIER's strengths are its rich facilities for expressing extraction rules for semi-structured text and for repetitive phrases in free-running text. Its weaknesses are that extraction rules must be hand-coded by experts at developing complex regular expressions and that no help is available for identifying patterns for which rules are needed (typically many dozen for a book like

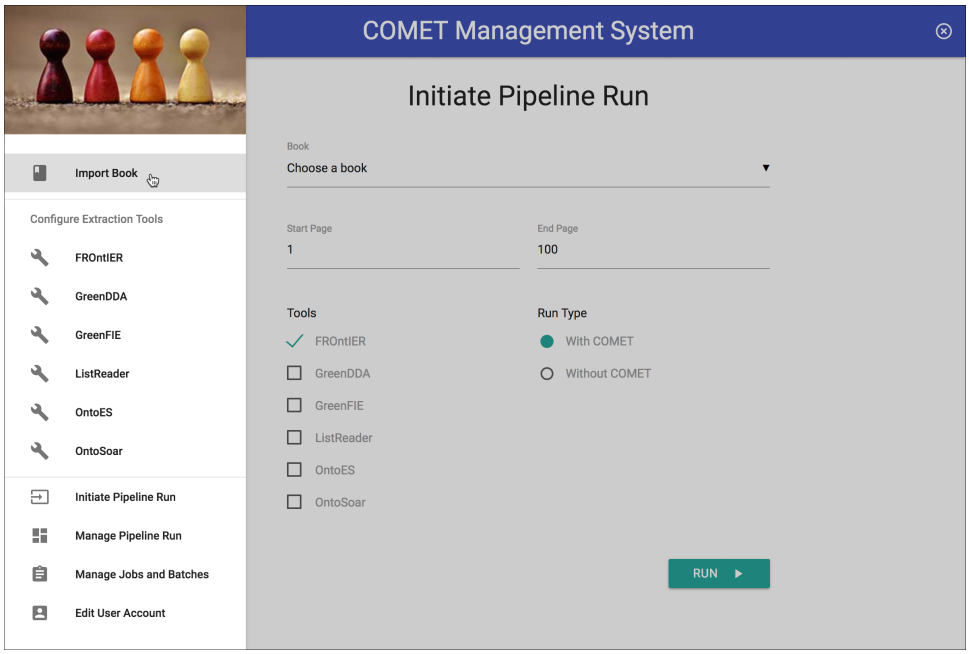



*Figure 26: Management System Interface*

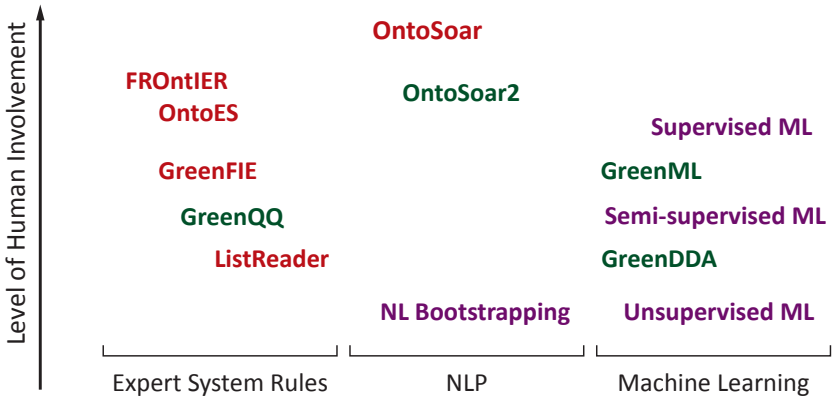
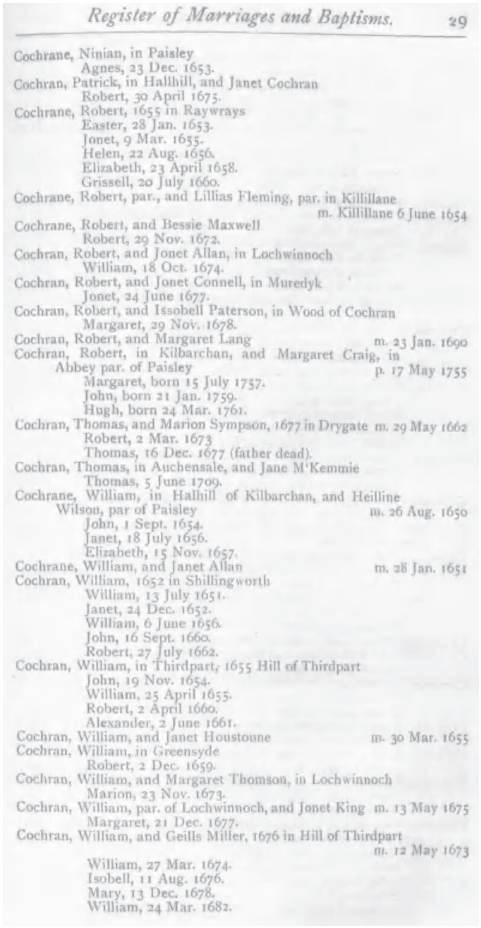


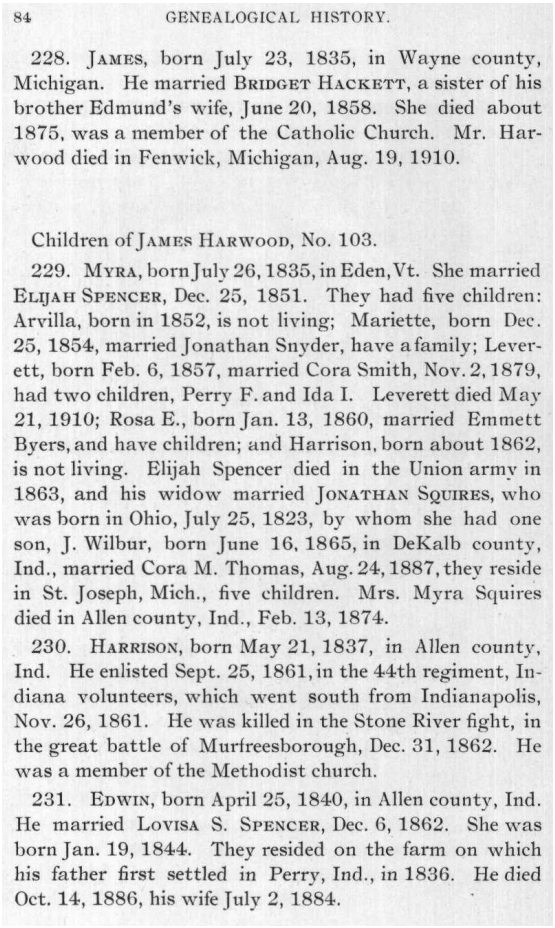

*Figure 27: Extraction Tools: Methodology Type and Ease of Use*



*Register of Marriages and Baptisms.* 29

Cochrane, Ninian, in Paisley
Agnes, 23 Dec. 1653.
Cochran, Patrick, in Hallhill, and Janet Cochran
Robert, 30 April 1675.
Cochrane, Robert, 1655 in Raywrays
Easter, 28 Jan. 1653.
Jonet, 9 Mar. 1655.
Helen, 22 Aug. 1656.
Elizabeth, 23 April 1658.
Grissell, 20 July 1660.
Cochrane, Robert, par., and Lillias Fleming, par. in Killillane
m. Killillane 6 June 1654
Cochrane, Robert, and Bessie Maxwell
Robert, 29 Nov. 1672.
Cochran, Robert, and Jonet Allan, in Lochwinnoch
William, 18 Oct. 1674.
Cochran, Robert, and Jonet Connell, in Muredyk
Jonet, 24 June 1677.
Cochran, Robert, and Issobell Paterson, in Wood of Cochran
Margaret, 29 Nov. 1678.
Cochran, Robert, and Margaret Lang m. 23 Jan. 1690
Cochran, Robert, in Kilbarchan, and Margaret Craig, in Abbey par. of Paisley p. 17 May 1755
Margaret, born 15 July 1757.
John, born 21 Jan. 1759.
Hugh, born 24 Mar. 1761.
Cochran, Thomas, and Marion Sympson, 1677 in Drygate m. 29 May 1662
Robert, 2 Mar. 1673
Thomas, 16 Dec. 1677 (father dead).
Cochran, Thomas, in Auchensale, and Jane M'Kemmie
Thomas, 5 June 1709.
Cochrane, William, in Halhill of Kilbarchan, and Heilline Wilson, par of Paisley m. 26 Aug. 1650
John, 1 Sept. 1654.
Janet, 18 July 1656.
Elizabeth, 15 Nov. 1657.
Cochrane, William, and Janet Allan m. 28 Jan. 1651
Cochran, William, 1652 in Shillingworth
William, 13 July 1651.
Janet, 24 Dec. 1652.
William, 6 June 1656.
John, 16 Sept. 1660.
Robert, 27 July 1662.
Cochran, William, in Thirdpart, 1655 Hill of Thirdpart
John, 19 Nov. 1654.
William, 25 April 1655.
Robert, 2 April 1660.
Alexander, 2 June 1661.
Cochran, William, and Janet Houstoune m. 30 Mar. 1655
Cochran, William, in Greensyde
Robert, 2 Dec. 1659.
Cochran, William, and Margaret Thomson, in Lochwinnoch
Marion, 23 Nov. 1673.
Cochran, William, par. of Lochwinnoch, and Jonet King m. 13 May 1675
Margaret, 21 Dec. 1677.
Cochran, William, and Geills Miller, 1676 in Hill of Thirdpart
m. 12 May 1673
William, 27 Mar. 1674.
Isobell, 11 Aug. 1676.
Mary, 13 Dec. 1678.
William, 24 Mar. 1682.

*Figure 28: Page 29 of the Kilbarchan Parish Record (Grant 1912)*



84 GENEALOGICAL HISTORY.

228. JAMES, born July 23, 1835, in Wayne county, Michigan. He married BRIDGET HACKETT, a sister of his brother Edmund's wife, June 20, 1858. She died about 1875, was a member of the Catholic Church. Mr. Harwood died in Fenwick, Michigan, Aug. 19, 1910.

Children of JAMES HARWOOD, No. 103.

229. MYRA, born July 26, 1835, in Eden, Vt. She married ELIJAH SPENCER, Dec. 25, 1851. They had five children: Arvilla, born in 1852, is not living; Mariette, born Dec. 25, 1854, married Jonathan Snyder, have a family; Leverett, born Feb. 6, 1857, married Cora Smith, Nov. 2, 1879, had two children, Perry F. and Ida I. Leverett died May 21, 1910; Rosa E., born Jan. 13, 1860, married Emmett Byers, and have children; and Harrison, born about 1862, is not living. Elijah Spencer died in the Union army in 1863, and his widow married JONATHAN SQUIRES, who was born in Ohio, July 25, 1823, by whom she had one son, J. Wilbur, born June 16, 1865, in DeKalb county, Ind., married Cora M. Thomas, Aug. 24, 1887, they reside in St. Joseph, Mich., five children. Mrs. Myra Squires died in Allen county, Ind., Feb. 13, 1874.

230. HARRISON, born May 21, 1837, in Allen county, Ind. He enlisted Sept. 25, 1861, in the 44th regiment, Indiana volunteers, which went south from Indianapolis, Nov. 26, 1861. He was killed in the Stone River fight, in the great battle of Murfreesborough, Dec. 31, 1862. He was a member of the Methodist church.

231. EDWIN, born April 25, 1840, in Allen county, Ind. He married LOVISA S. SPENCER, Dec. 6, 1862. She was born Jan. 19, 1844. They resided on the farm on which his father first settled in Perry, Ind., in 1836. He died Oct. 14, 1886, his wife July 2, 1884.

*Figure 29: Page 84 of the History of the Harwood Families (Harwood 1911)*

*The Ely Ancestry*). Painstakingly finding all the patterns, or at least sufficient to make the rule-set worthwhile, and developing a non-conflicting and reasonably minimal set of rules can be tedious and time-consuming.

*OntoES*

As explained in Section 3.2, OntoES adopts FROntIER's ontology-snippet extraction capability and extends it for complex annotations. Further, for our genealogy application, OntoES specifically targets its extraction to the three ontologies in Figures 7, 8, and 9, and will be supported by the extraction rule creation and testing interface illustrated in Figure 10.

In an evaluation of OntoES, we created 25 ontology-snippet extraction rules—7 for the Person ontology, 4 for Couple, and 14 for Family—which together were sufficient to capture all the information of interest from the Kilbarchan page in Figure 28 and also the previous and subsequent page. Then, in a fully automatic extraction run over the 143 pages of *The Kilbarchan Parish Record*, these 25 rules extracted information for 8,539 individuals. Based on a check of several randomly chosen pages, the automatic extraction's F-score was judged to be near 95%.

Strengths of OntoES include its form-filling paradigm and its support for ontology-snippet extraction rule creation. Its weakness, like FROntIER's, is its requirement for hand-coded extraction rules. However, an interface like the one in

Figure 10 nicely supports rule creation and testing, and goes a long way to mitigate the problems associated with hand-coding extraction rules.

*GreenFIE*

As explained in Section 3.3, GreenFIE synergistically works with users who are filling in forms. Once a record's fields are filled in, GreenFIE can generate a regular-expression extraction rule that would extract the same information, generalize the rule, execute it, and populate subsequent records.

Table 2 gives the results of an experiment we conducted. For both *The Ely Ancestry* and *The Kilbarchan Parish Record* we selected a sequence of three pages and filled in records until we achieved 100% recall. GreenFIE generated an extraction rule for every record filled in by the user and executed it to fill in subsequent records for the user to check and in some cases complete if GreenFIE filled in only part of the record. Although incorrectly filled-in records could have been deleted, for the experiment none were deleted so that we could measure the precision of the GreenFIE-generated extraction rules.

The rate of task completion is the total number of records correctly extracted divided by the total number of user actions (new records filled-in and partially filled-in records completed). The maximum rate, for example, is 19.60, which tells us that filling in just five records is sufficient to extract all the information in the 98 records for the Person form on the three Kilbarchan pages. Table 2 also gives the total number of records extracted, 333, and the total number of user actions, 121. We estimated the number of data values extracted in these 333 records to be about 1,000 so that approximately 8.2 data values were obtained for each user-extracted or user-edited record.

As strengths, we note that GreenFIE users never need to write, or even edit, extraction rules. We also note that GreenFIE's rules are highly precise (0.99 for the overall precision in Table 2), which when coupled with COMET's highlighting of extracted text for records greatly facilitates check and correct. Considering weaknesses, we were disappointed with the overall rate of task completion. Although quite precise, GreenFIE's heuristic generalization of regular-expression rules will likely never rival a human expert. Human experts may, however, benefit from having GreenFIE generate regular expressions for them to adjust rather than having to write rules manually.

*ListReader*

As explained in Section 3.5, ListReader discovers record patterns in text and generates extraction rules to recognize these records. To extract the information from recognized records, a human provides labels for the component parts of the recognized text to map each part to an ontology.

Table 3 shows the results of an experiment we conducted. We developed ListReader using *The Ely Ancestry* and tested it on a similar book, *Shaver-Dougherty* (Shaffer 1997), and on *The Kilbarchan Parish Record*. For the ground truth, we labeled a few dozen pages in these books. The Shaver-Dougherty results in Table 3 were essentially obtained after labeling about 25 patterns, and the Kilbarchan after only about a half dozen—"essentially" because both books have a long tail of less frequent patterns, many of which were labeled.

As strengths, ListReader processes an entire book at once, discovering patterns without human intervention, and it efficiently assists users with the labeling task by ordering the patterns by greatest impact first and by cross-labeling, which sometimes obviates the need to label a pattern at all. ListReader's weaknesses include its inability to recognize large patterns such as parents-with-child lists and its apparent low recall as seen in the experimental results.

*OntoSoar*

As explained in Section 3.7 OntoSoar parses text and then applies a cognitive reasoner to map the resulting parse to an ontology (Lindes et al. 2015). The results of running OntoSoar over a text snippet from Figure 1 are in Table 4 and over a text snippet from Figure 29 are in Table 5. Recall errors are mostly due to information expressed in linguistic patterns—especially for list-

*Table 2: GreenFIE Results*

| | User Action | Correct (=Total) | Correct per User Action | Incorrect | Precision | Recall | F-score |
|---|---|---|---|---|---|---|---|
| Ely | 86 | 157 | 1.83 | 2 | 0.99 | 1.00 | 0.99 |
| Person | 28 | 76 | 2.71 | 2 | 0.97 | 1.00 | 0.99 |
| Couple | 26 | 43 | 1.65 | 0 | 1.00 | 1.00 | 1.00 |
| Family | 32 | 38 | 1.19 | 0 | 1.00 | 1.00 | 1.00 |
| Kilbarchan | 35 | 176 | 5.03 | 3 | 0.98 | 1.00 | 0.99 |
| Person | 5 | 98 | 19.60 | 0 | 1.00 | 1.00 | 1.00 |
| Couple | 15 | 30 | 2.00 | 2 | 0.94 | 1.00 | 0.97 |
| Family | 15 | 48 | 3.20 | 1 | 0.98 | 1.00 | 0.99 |
| Overall | 121 | 333 | 2.75 | 5 | 0.99 | 1.00 | 0.99 |

ing children—that have not yet been encoded into the system.

As strengths, OntoSoar works on free running text, and it can process semi-structured text as well. The LG-parser it uses does not depend on the chunks of text it processes being complete sentences nor on the phrases being grammatically correct. OntoSoar's main weakness is its dependence on hand-coded production rules for Soar (Laird 2012). Obtaining the results in Tables 4 and 5 required 240 hand-coded production rules.

### 8.1.2 Academic Work in Progress

Preliminary results for the extraction tools we present in this section look promising, but much more academic work is needed to bring these tools to fruition and ready for tech-transfer to a production environment.

*GreenQQ*

Developed in an independent effort, GreenQQ is currently being investigated for adaption and use in our FamilySearch reading system. As explained in Section 3.3, it interacts with users via text-snippet examples. Beginning with the most frequently occurring token-sequence patterns that contain information of interest, GreenQQ proposes extraction rules for a user to adjust and accept for execution. After execution, GreenQQ again proposes rules and continues in this synergistic interaction cycle until the results are satisfactory.

In an initial trial run, we applied GreenQQ to the 119 pages of *The Kilbarchan Parish Record* that are similar to the page in Figure 28. Interacting with a user for less than 50 minutes, GreenQQ created 40 templates that classified sequences of tokens into 5 classes (HEAD of household, WIFE, BABY, GEO location, and DATE of birth, christening, marriage, or proclamation of marriage). In the final cycle, GreenQQ processed 9,464 lines of text with 89,391 tokens and found 17,206 matches in 5 seconds of runtime. Table 6 shows the accuracy results of the extraction for three randomly chosen pages (Embley and Nagy 2017).

A strength of GreenQQ is its ease of use. Classified text snippets presented to users as candidate extraction rules require only that a user is knowledgeable enough to identify and classify the part of the text snippet to be extracted. A weakness of GreenQQ is that it only does named entity recognition (NER). Thus, we must restrict GreenQQ's usage to ontologies with one central non-lexical object set that is directly connected to all of its lexical object sets through relationship sets—like the ontologies in Figures 7, 8, and 9. Furthermore, in its currently implemented state, an additional weakness is that it cannot be used when ontology records are interleaved as *Couple* and *Family* records are in Figure 1. However, as explained in Section 3.3, we expect to be able to resolve this weakness. Resolving it may require human input, which could lessen GreenQQ's ease of use.

*GreenML/GreenDDA*

We have begun to carry out experiments using the different levels of machine learning discussed in Section 3.6. In a preliminary test we used the

*Table 3: ListReader Results*

| | Prec. | Rec. | F-score |
|---|---|---|---|
| Shaver-Dougherty | 0.94 | 0.39 | 0.55 |
| Kilbarchan | 0.94 | 0.54 | 0.68 |

*Table 4: OntoSoar Results for the Charles Christopher Lathrop Family and the First Three Lines of the Commentary about Miss Emma Goble Lathrop in Figure 1*

| Category | Exists | Found | Correct | P Errors | R Errors | P | R | F |
|---|---|---|---|---|---|---|---|---|
| Persons | 12 | 11 | 11 | 0 | 1 | 100.0% | 91.7% | 95.7% |
| Births | 6 | 6 | 6 | 0 | 0 | 100.0% | 100.0% | 100.0% |
| Deaths | 4 | 4 | 4 | 0 | 0 | 100.0% | 100.0% | 100.0% |
| Marriages | 1 | 1 | 1 | 0 | 0 | 100.0% | 100.0% | 100.0% |
| Sons & Daughters | 7 | 2 | 2 | 0 | 5 | 100.0% | 28.6% | 44.4% |
| Totals/Average | 30 | 24 | 24 | 0 | 6 | 100.0% | 80.0% | 88.9% |

*Table 5: OntoSoar Results for Paragraph 229 Plus the Header Preceding the Paragraph in Figure 29*

| Category | Exists | Found | Correct | P Errors | R Errors | P | R | F |
|---|---|---|---|---|---|---|---|---|
| Persons | 19 | 15 | 14 | 1 | 4 | 93.3% | 73.7% | 82.4% |
| Births | 8 | 8 | 7 | 1 | 0 | 87.5% | 87.5% | 87.5% |
| Deaths | 5 | 3 | 3 | 0 | 2 | 100.0% | 60.0% | 75.0% |
| Marriages | 6 | 6 | 4 | 2 | 0 | 66.7% | 66.7% | 66.7% |
| Sons & Daughters | 9 | 0 | 0 | 0 | 9 | N/A | 0.0% | 0.0% |
| Totals/Average | 47 | 32 | 28 | 4 | 15 | 87.5% | 59.6% | 70.9% |

*Table 6: GreenQQ Results*

| | | | | | Soft:Correct=Correct+Partial | | | Hard:Incorrect=Partial+Incorrect | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Class | Total | Correct | Partial | Incorrect | Recall | Precision | F-score | Recall | Precision | F-score |
| HEAD | 72 | 71 | 0 | 0 | 0.99 | 1.00 | 0.99 | 0.99 | 1.00 | 0.99 |
| WIFE | 56 | 53 | 1 | 0 | 0.96 | 1.00 | 0.98 | 0.95 | 0.98 | 0.96 |
| BABY | 92 | 91 | 0 | 1 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| DATE | 123 | 116 | 0 | 0 | 0.94 | 1.00 | 0.97 | 0.94 | 1.00 | 0.97 |
| GEO | 65 | 63 | 0 | 4 | 0.97 | 0.94 | 0.95 | 0.97 | 0.94 | 0.95 |
| Overall | 408 | 394 | 1 | 5 | 0.97 | 0.99 | 0.98 | 0.97 | 0.99 | 0.98 |

An extract was judged to be partially correct if the sequence of labeled tokens for the extract was a proper subsequence of the ground-truth token sequence for the extract. Soft scoring counted partially correct extracts as being correct, while Hard scoring counted them as being incorrect.

Stanford CoreNLP system (Stanford CoreNLP n.d.) in OTS (off-the-shelf), GreenML, and GreenDDA settings to perform human-supervised annotation of a typical 10-page range of *The Ely Ancestry*. Figure 30 displays the results.

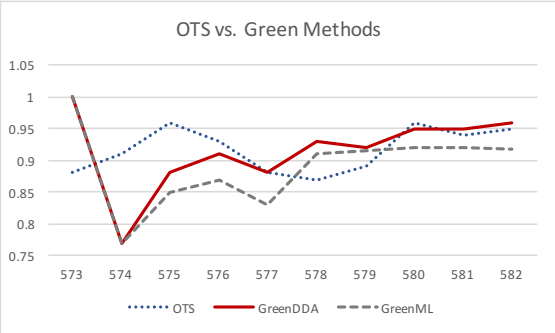



*Figure 30: OTS vs. Green Methods: F-measure Annotation Results for a 10-page Range*

For the green techniques, we annotated according to the following general scenario:

1. A user initiates the process by hand-annotating the first page in a book.
2. A (so-far small) machine learning model is trained on that gold standard, and it serves to annotate the next page. The user then corrects that page, which is added to the gold standard, and the process repeats.
3. After some number of these iterations, from at least one to some empirically-determined number, the user stops hand-annotating. Call this the transition point.

From the transition point, the two green methods differ:

- GreenML builds the final user-supervised model and then proceeds to annotate the rest of the book without further user intervention or learning.
- GreenDDA, after building the final user-supervised model, proceeds to annotate the rest of the book alone. However, after each page, its raw annotations are added incrementally to the entire training set and a new model is trained.

To assess and illustrate the difference in performance in these two approaches, we swept across all possible user involvement scenarios in the 10-page range. For each method we averaged performance (measured by F-measure) across all possible transition points.

OTS annotation performance varies substantially by page, given the variability of the data across pages. GreenML gradually decreases as more pages are encountered, reflecting the static nature of the trained model past the transition point. After the eighth page, the other two approaches perform better than GreenML. However, GreenDDA—after the fifth page—generally outperforms OTS on average. This is because training past the transition point occurs after every page, albeit with raw annotations. With such a high F-measure (over 0.9), the results (even after user supervision) are reasonable enough to yield better models over time. Whether these NER results will hold for other page sequences in the book or in other books and whether results for relationship recognition will be similar remains to be seen.

*OntoSoar2*

Writing Soar production rules by hand is a complex and intensive process, calling into question the scalability of the current OntoSoar approach. Each new type of linguistic construction that contains information that could map to the semantic representation and ultimately the ontology needs to be treated in this manner. A future instantiation of the system, OntoSoar2, could include a rule compiler which would allow for rules to be coded in a metalinguistic grammar and compiled directly into Soar code, as is being done elsewhere (Lindes and Laird 2016).

Another potential feature for OntoSoar2 would be an improved syntactic parsing process. The current LG parser, while robust and flexible, also requires hand-coding of custom rules, and does not include any machine learning functionality. Dependency parsers (Mel'čuk 1988) have been implemented for many languages beyond English in frameworks such as CoreNLP, which support machine learning and incremental training.

Furthermore, it is unclear how flexible the current OntoSoar semantic representation framework (Embodied Construction Grammar) is for supporting large-scale NLP applications like ours (Lindes et al. 2017). While it has proven effective in relatively narrow domains such as robotics (Lindes and Laird 2017), OntoSoar2 work would need to explore the appropriateness of this or other alternative deep semantic representations.

Finally, OntoSoar2 has the potential to further implement of some of the language processing capabilities inherent in the current cognitive architecture framework. This could include such "human" aspects of processing such as being incremental (i.e. word-by-word rather than a sentence as a whole); eclectic (i.e. leveraging more pragmatic and real-world knowledge); repair-based (i.e. reformulating hypotheses that prove untenable in the presence of further context); and grounded (i.e. integrating perceptual input such as page layouts).

```
**********************************
Person osmx190: ALBRIGHT, ESTHER R.
**********************************
Name:
   Conclusion: Esther R. Albright
   Interpreted Document Text: ALBRIGHT, ESTHER R.
   Original Document Text: ALBRIGHT, ESTHER R.
   Inferred Formal Name: Esther R. Morris Albright
      Title:
      First Names: Esther R.
      Last Names: Morris Albright
      Suffix:
   Inferred Birth Name: Esther R. Morris
   Inferred Married Name: Esther R. Albright
   Gender: Female
Facts:
   BirthDate:
      Conclusion: 22 July 1863
      Interpreted Document Text: 22 July 1863
      Original Document Text: 22 July 1863
   BirthPlace:
      Interpreted Document Text: Butler Co OH
   DeathDate:
      Conclusion: 1 January 1946
      Interpreted Document Text: 1 Jan 1946
      Original Document Text: 1 Jan 1946
   DeathPlace:
      Interpreted Document Text: 113 Sherman St Dayton OH
   BurialDate:
      Conclusion: 3 January 1946
      Interpreted Document Text: 3 Jan 1946
      Original Document Text: 3 Jan 1946
   BurialPlace:
      Interpreted Document Text: Abbottsville Cem Dke Co OH
Marriage Relationships:
   Spouse: osmx334 (l-linfield S. Albright)
ParentOf Relationships
ChildOf Relationships:
   osmx169 (Thomas Benton Morris)
   osmx480 (Angeline Harrod)
```

*Figure 31: Esther Albright Information as a Result of Full Pipeline Processing*

## 8.2 Pipeline Processing

Given a book that has been scanned and OCR'd, the pipeline processes the book from import (a single PDF document of the full book) to export (a GedcomX document for each page that contains genealogical information). The pipeline runs in several steps:

1. *Prepare Pages*. Split the input PDF document into pages. For each PDF page generate four additional files: an xml file containing the OCR information for the page; a text file of the OCR'd characters assembled into words and lines; and a PNG image and HTML document that together let a COMET user view and work with the page as an image superimposed over hidden OCR'd text aligned with the text in the image.
2. *Extract Information*. Apply the extraction tools to the text files and, for each tool and page, populate the ontology in Figure 15.
3. *Merge Information*. For each page merge the information in the tool-populated ontologies and create a single populated ontology.
4. *Check and Correct Information*. Run a constraint checker over the extracted and merged information, discover anomalies, and fix those identified as being automatically rectifiable. If patrons are to check and correct the information, split the information into *Person*, *Couple*, and *Family* form data and for each form/page combination display the filled in form in COMET for a user to check and correct.
5. *Enhance Information*. To the extent possible, standardize person names, place names, and dates, and infer gender, birth names, married names, and formal names.
6. *Generate GedcomX*. Generate a GedcomX file for each page containing genealogical information. In addition, for each person having associated genealogical information, generate a person information document detailing associated names, event dates and places, and marriage and parent-child relationships (e.g. see Figure 31). Generate also an HTML document with all the extracted information for a person

highlighted on an image of the page (or pages, if the information spans more than one page).

A prototype of the pipeline runs from beginning to end, and the code is being improved as we gain experience and encounter new edge cases. (Pipeline processing for complex annotations, for example, is currently being added.) The extraction engines, whose academic prototypes are complete, all run, but considerable work is still required to convert them into tools usable by anyone besides ourselves. Academic research is continuing for extraction tools still under development. COMET has been used by subjects in some experimental evaluations; they generally find it usable after a few minutes of training (Woodfield et al. 2016). We have only begun to build a management system that will control the processing of books through the pipeline.

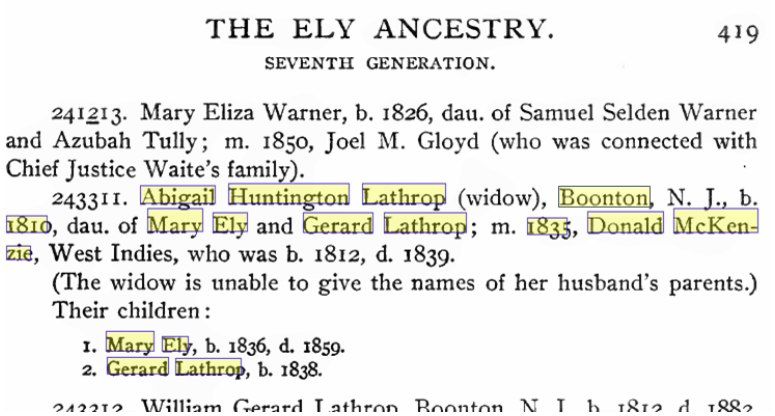
### 8.3 Ingest into *Family Tree*

We have conducted several field tests to determine how our extraction results can contribute to *Family Tree* (Embley et al. 2017).

**Ely** We processed the page in Figure 1 through the pipeline—extracted information using FROntIER, Ontos, and OntoSoar; merged it; checked and corrected it with COMET; standardized the data; inferred gender and extended name information; and generated person information documents for each person mention on the page having associated genealogical information. To compare the effort to ingest information manually with a proposed automatic ingest, we updated *Family Tree* by hand according to the generated person information documents. We filled in search forms, identified matching *Family Tree* records, merged duplicates (if any), checked the matching records, and added to them source documentation and missing information. From 31 unique generated person information documents, we found that 28 matched exactly one *Family Tree* person record. For the Mary Ely married to Gerard Lathrop we found two, as Figure 21 shows, and we merged them. Donald McKenzie's and Abigail Huntington Lathrop's person information documents both matched three records that were themselves duplicates, and in both cases we merged the three records. We added highlighted source documents like the one in Figure 32 for all 31 matched tree records. Overall, we (1) replaced two primary names with more complete names (e.g. "Emma Sutherland Goble" in place of "Emma S. Goble"); (2) replaced six uncertain BMD (Birth/Marriage/Death) facts (e.g. "about 1831" or merely "deceased") with certain facts; (3) added two missing BMD facts, and (4) added eight supplementary facts such as married names or alternate spellings of names. All of this work, which could have been done fully automatically within seconds of compute time, took more than five hours of tedious typing, checking, clicking, and waiting for responses from the FamilySearch web site.

**Kilbarchan** In a fully automatic extraction run over the 143-pages of the Kilbarchan parish record (Grant 1912), the pipeline, running without COMET-user intervention, created 8,539 person information documents like the one in Figure 31. The F-score for the automatic extraction was judged to be near 95%. In a sample of 150 of these 8,539 person information documents, a prototype matching algorithm was 100% correct when the tools extracted eight or more distinct items of information for the person. Of the correctly matched person records, 20% had information that could be added to *Family Tree*, including adding or fixing first and last names, event dates, and parent-child relationships.

**Miller** Similar to our Kilbarchan field test, in a fully automatic extraction run over the 396-page Miller book (Miller Funeral Home Records, 1917 – 1950, Greenville, Ohio 1990), information for 12,226 individuals was extracted. Of the 1,280 individuals the matching algorithm found in *Family Tree* with certainty, the Miller records provided information that could be added to 57% of them.



THE ELY ANCESTRY. 419

SEVENTH GENERATION.

241213. Mary Eliza Warner, b. 1826, dau. of Samuel Selden Warner and Azubah Tully; m. 1850, Joel M. Gloyd (who was connected with Chief Justice Waite's family).

243311. Abigail Huntington Lathrop (widow), Boonton, N. J., b. 1810, dau. of Mary Ely and Gerard Lathrop; m. 1833, Donald McKenzie, West Indies, who was b. 1812, d. 1839.

(The widow is unable to give the names of her husband's parents.)

Their children:

1. Mary Ely, b. 1836, d. 1859.
2. Gerard Lathrop, b. 1838.

243312. William Gerard Lathrop, Boonton, N. J., b. 1813, d. 1882.

*Figure 32: Abigail's Information Highlighted*

## 8.4 Web of Knowledge

The grand challenge of collecting and representing the world knowledge of any domain (scientific, geo-political, or any other) in a structured, searchable, online repository has been and is the dream of many visionaries. Projects with this vision in mind include: open information extraction systems that have extracted billions of assertions as the basis for both common-sense knowledge and novel question-answering systems (Banko et al. 2007; Etzioni 2011; Fader et al. 2011; Mausam et al. 2012); Yahoo!'s Web of Concepts (Dalvi et al. 2009); and large projects such as Cyc (Lenat and Guha 1989), Freebase (Bollacker et al. 2008), DBpedia (Auer et al. 2007), YAGO (Suchanek et al. 2007), YAGO3 (Mahdisoltani et al. 2015), and NELL (Mitchell et al. 2015).

A Web of Knowledge (WoK), as we envision it (Embley et al. 2011), aims at a specific domain of interest such as family history. The backbone of each WoK is an ontology that describes the domain and is to be populated with information. As an example, FamilySearch's conceptualization of family history is populated with basic genealogical information for about 5.8 billion persons, 1.2 billion of which are on the public *Family Tree*. In addition, it stores over a billion auxiliary data items including more than 830 million images and transcripts of documents used as source documentation of genealogical facts; more than 22 million stories and pictures submitted by individuals as memories of their ancestors; more than 350,000 family history books with pages like those in Figures 1, 28, and 29; and more than 2,000 historical record collections (e.g. census records, burial records, military records) consisting of more than a billion historical documents, many millions of which have been indexed for semantic search (FamilySearch Company Facts 2017).

The backbone of a WoK should be an ontology in its true sense—domain knowledge, agreed upon and shared by a community (Dillon et al. 2008; Gruber 1993). A WoK ontology is itself worthy of study and understanding and may evolve over time as its community comes to a greater understanding of itself. When populated, the ontology may more rightly be seen as a *knowledge base*—a particular state of the domain whose information can be queried and updated (Dillon et al. 2008). The extraction ontologies we have been discussing in this paper may more accurately be seen as *operational views* of the larger WoK ontology—*views* in the traditional database sense and thus themselves ontologies albeit for a much smaller domain and *operational* in the sense that they enable document reading and query search along with computational operations over stored data.

We focus our remaining comments on WoK query formulation and execution and show its connection to ontological document reading. The correspondence among form, ontology, and semi-structured text provides insight into how to query

the WoK: (1) create a form for the results wanted and have the system learn how to correlate it with the WoK ontology and fill it in (Embley 1989); (2) explore the graphical conceptual model view of the WoK ontology and filter desired information directly from the graphical view (Czejdo et al. 1990); (3) generate forms from WoK ontology snippet views and allow constraints to be applied to form fields (Zitzelberger et al. 2015); and (4) from a free-form query "read" and "understand" it to obtain what is wanted (Zitzelberger et al. 2015).

Most interesting from the standpoint of document reading are HyKSS free-form queries (Zitzelberger et al. 2015). "HyKSS" which stands for "Hybrid Keyword and Semantic Search" allows users to issue free-form queries such as "Find Abigail McKenzie who lived to be nearly 100 years old and whose husband was from the West Indies" HyKSS applies its collection of extraction ontologies to the query. Here, the *Name* recognizer in the ontology in Figure 2 recognizes "Abigail McKenzie" as a name, and "West Indies" is recognized as a *LocationName* in the ontology in Figure 22. The phrase "lived to be nearly 100 years old" associates with an *Age* operator in a *Date* data frame, and "husband" associates with the marriage relationship. The HyKSS keyword recognizer ignores stop words ("Find", "who", "and", "whose", "was", "from", "the"), words recognized as denoting operators ("lived to be", "years old"), and non-equality operands ("nearly 100") and treats the remaining words and any quoted phrases as keywords ("Abigail", "McKenzie", "West", "Indies"). As is the case for all search engines, keywords in source documents have been previously indexed. For HyKSS, semantics have also been indexed by applying extraction ontologies to documents, in particular for this example the page in Figure 1. Once the query has been "read" and "understood" (i.e. once the information and keywords have been extracted from the query and mapped to an ontology), a formal query can be generated and executed (Zitzelberger et al. 2015). The results are returned in search-engine fashion—a ranked clickable list of URLs with a snippet showing what was matched. In our example, the first link would likely lead to the page in Figure 1. Clicking would bring up the page with the identified keywords and semantic data highlighted.

Although processing free-form queries is interesting from a document-reading point of view, in applications like genealogy in which the ontological conceptualizations are well known, starting with a form search is likely better. Figure 33 shows one of FamilySearch's forms mocked-up with an additional search field for *Children* so that it better matches the underlying ontology and also an additional search field for *Keywords* to allow for HyKSS-like search over a hybrid of keywords and semantics. The form is filled in for the information extracted from Figure 1 for Abigail Huntington Lathrop with "Boonton" and "New Jersey" as keywords. The results should come back in a search-engine-like list of links to documents, which when clicked should yield a highlighted document like the one in Figure 32.

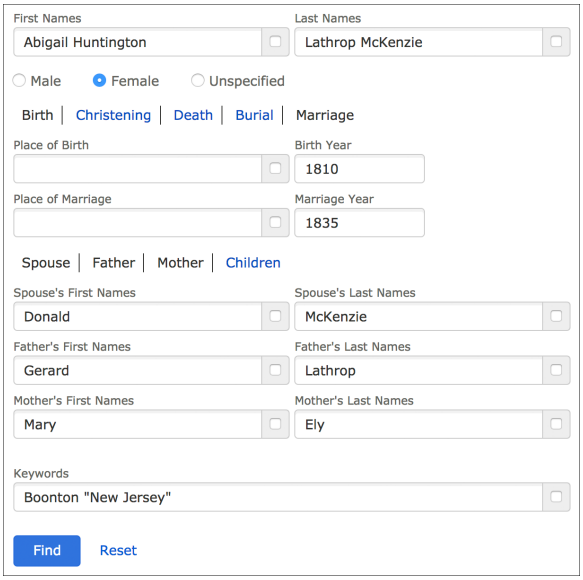



*Figure 33: Query for Abigail Huntington Lathrop McKenzie*

Finally, we mention that for a world-wide application like FamilySearch, query processing should be multilingual. We show in (Embley et al. 2014) how extraction ontologies can form the backbone of a query processing system that allows queries to be expressed in a user's native language,

then processed in the language of the document, with results translated back into the user's native language. Clicking on resulting document links would bring up highlighted original-language documents.

## 9 Conclusion

We have defined ontology-based document reading and have expounded upon our experience in implementing an ontological document reading system. The reading system transforms asserted facts stated in a document into objects and relationships and populates the object and relationship sets of a conceptual model. It thus populates the ontology represented by the conceptual model, which gives meaning to the extracted text. Lexical object sets are populated directly with text tokens found in the document. Non-lexical object sets are populated by ontological commitment.

Besides directly populating lexical object sets with text found in a document, a reading system should also be able to "read between the lines" and infer author-implied facts such as a female's married name given her spouse's name or a birth name given a child's father's name. The populating objects and relationships must also make sense with respect to declared ontological constraints or otherwise be rejected, fixed, or at least questioned. An ideal reading system should also be able to extend or adjust its ontological conceptualizations and make connections among them.

In this experience report, we have described an implemented ontology-based document reading system for contributing to FamilySearch's online wiki-like *Family Tree*. The reading system reads an input document, usually a book, and populates the target extraction ontology in Figure 2 with information extracted from the document's text by an ensemble of extraction engines. It then merges and checks the information against ontological constraints and corrects constraint violations when possible. Optionally, it allows a human to check and and correct extraction results. Given the extraction results, it standardizes names and dates and infers gender and various name forms. Finally, it generates both a GedcomX document for each book page and an individual report for each person mentioned in the book who has genealogical information. Either the GedcomX documents or the individual reports can be used for automatic or user-checked semi-automatic import into *Family Tree*.

The document reading system we have elucidated works particularly well for semi-structured document reading and for populating ontologies with facts within rich but narrow domains of interest like family history. We nevertheless foresee the possibility of using ontology-based document reading as a means to contribute to the construction of a general Web of Knowledge, possibly involving many interconnected domains of interest.

## References

Aristotle (about 350BC) Metaphysics. (1993 translation). Oxford University Press, New York

Auer S., Bizer C., Kobilarov G., Lehmann J., Cyganiak R., Ives Z. (2007) DBpedia: A Nucleus for a Web of Open Data. In: Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference. Busan, Korea, pp. 722–735

Banko M., Cafarella M., Soderland S., Broadhead M., Etzioni O. (2007) Open Information Extraction from the Web. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007). Hyderabad, India, pp. 2670–2676

Benjelloun O., Garcia-Molina H., Menestrina D., Su Q., Whang S. E., Widom J. (2009) Swoosh: A Generic Approach to Entity Resolution. In: The VLDB Journal—The International Journal on Very Large Data Bases 18(1), pp. 255–276

Bhattacharya I., Getoor L. (2007) Collective Entity Resolution in Relational Data. In: ACM Transactions on Knowledge Discovery from Data 1(1), pp. 1–36

Bollacker K., Evans C., Paritosh P., Sturge T., Taylor J. (2008) Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver, Canada, pp. 1247–1250

Buitelaar P., Cimiano P., Frank A., Hartung M., Racioppa S. (2008) Ontology-based Information Extraction and Integration from Heterogeneous Data Sources. In: International Journal of Human Computer Studies 66(11), pp. 759–788

Buitelaar P., Cimiano P., Haase P., Sintek M. (2009) Towards Linguistically Grounded Ontologies. In: Proceedings of the 6th European Semantic Web Conference (ESWC'09). Heraklion, Greece, pp. 111–125

Chen P. P. (1983) English Sentence Structure and Entity-Relationship Diagrams. In: Information Sciences 29(2–3), pp. 127–149

Chiticariu L., Li Y., Reiss F. (2013) Rule-based Information Extraction is Dead! Long Live Rule-based Information Extraction Systems! In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle, Washington, pp. 827–832

Cimiano P. (2006) Ontology Learning and Population from Text: Algorithms, Evaluation and Applications. Springer, New York, New York

Cimiano P., Völker J. (2005) Text2Onto—A Framework for Ontology Learning and Data-driven Change Discovery. In: Proceedings of the 10th International Conference on Applications of Natural Language to Information Systems (NLDB'05). Alicante, Spain, pp. 227–238

Cimiano P., Völker J., Studer R. (2006) Ontologies on Demand? - A Description of the State-of-the-Art, Applications, Challenges and Trends for Ontology Learning from Text. In: 57, pp. 315–320

Clark K., Manning C. D. (2016) Improving Coreference Resolution by Learning Entity-level Distributed Representations. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016), pp. 643–653

Stanford CoreNLP. https://stanfordnlp.github.io-/CoreNLP/

Curland M., Halpin T. A. (2012) Enhanced Verbalization of ORM Models. In: On the Move to Meaningful Internet Systems: OTM 2012 Workshops, Confederated International Workshops: OTM Academy, Industry Case Studies Program, EI2N, INBAST, META4eS, OnToContent, ORM, SeDeS, SINCOM, and SOMOCO 2012, Rome, Italy, September 10-14, 2012. Proceedings, pp. 399–408

Czejdo B., Elmasri R., Embley D., Rusinkiewicz M. (1990) A Graphical Data Manipulation Language for an Extended Entity-Relationship Model. In: Computer 23(3), pp. 26–36

Dalvi N., Kumar R., Pang B., Ramakrishnan R., Tomkins A., Bohannon P., Keerthi S., Merugu S. (2009) A Web of Concepts. In: Proceedings of PODS'09. Providence, Rhode Island, pp. 1–12

Datalog User Manual. http://www.ccs.neu.edu /home/ramsdell/tools/datalog/datalog.html

Dillon T., Chang E., Hadzic M., Wongthongtham P. (2008) Differentiating Conceptual Modelling from Data Modelling, Knowledge Modelling and Ontology Modelling and a Notation for Ontology Modelling. In: Proceedings of the Fifth Asia-Pacific Conference on Conceptual Modelling. Wollongong, New South Wales, Australia, pp. 7–17

Duke: Fast Deduplication Engine. https://code.google.com/p/duke/

Eikvil L. (1999) Information Extraction from World Wide Web: A Survey. Tech Report No. 945. Norweigan Computing Center

Elmagarmid A., Ipeirotis P., Verykios V. (2007) Duplicate Record Detection: A Survey. In: IEEE Transactions on Knowledge and Data Engineering 18(1), pp. 1–16

Embley D. (1980) Programming With Data Frames for Everyday Data Items. In: Proceedings of the 1980 National Computer Conference. Anaheim, California, pp. 301–305

Embley D. (1989) NFQL: The Natural Forms Query Language. In: ACM Transactions on Database Systems 14(2), pp. 168–211

Embley D. (1998) Object Database Development: Concepts and Principles. Addison-Wesley, Reading, Massachusetts

Embley D., Campbell D., Jiang Y., Liddle S., Lonsdale D., Ng Y.-K., Smith R. (1999a) Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages. In: Data & Knowledge Engineering 31(3), pp. 227–251

Embley D., Campbell D., Jiang Y., Ng Y.-K., Smith R., Liddle S., Quass D. (1998a) A Conceptual-Modeling Approach to Extracting Data from the Web. In: Proceedings of the 17th International Conference on Conceptual Modeling (ER'98). Singapore, pp. 78–91

Embley D., Campbell D., Liddle S., Smith R. (1998b) Ontology-Based Extraction and Structuring of Information from Data-Rich Unstructured Documents. In: Proceedings of the 7th International Conference on Information and Knowledge Management (CIKM'98). Washington D.C., pp. 52–59

Embley D., Jackman D., Xu L. (2001) Multifaceted Exploitation of Metadata for Attribute Match Discovery in Information Integration. In: Proceedings of the International Workshop on Information Integration on the Web (WIIW'01). Rio de Janeiro, Brazil, pp. 110–117

Embley D., Jiang Y., Ng Y.-K. (1999b) Record-Boundary Discovery in Web Documents. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data (SIGMOD'99). Philadelphia, Pennsylvania, pp. 467–478

Embley D., Kurtz B., Woodfield S. (1992) Object-oriented Systems Analysis: A Model-Driven Approach. Prentice Hall, Englewood Cliffs, New Jersey

Embley D., Liddle S., Eastmond T., Lonsdale D., Price J., Woodfield S. (2017) Conceptual Modeling in Accelerating Information Ingest into *Family Tree*. In: Cabot J., Gómez C., Pastor O., Sancho M. (eds.) Conceptual Modeling Perspectives. Springer, Cham, Switzerland, pp. 69–84

Embley D., Liddle S., Lonsdale D. (2011) Conceptual Modeling Foundations for a Web of Knowledge. In: Embley D., Thalheim B. (eds.) Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges. Springer, Heidelberg, Germany, chap. 15, pp. 477–516

Embley D., Liddle S., Lonsdale D., Tijerino Y. (2014) Multilingual Extraction Ontologies. In: Buitelaar P., Cimiano P. (eds.) Towards the Multilingual Semantic Web. Springer, pp. 155–173

Embley D., Liddle S., Park J. (2016) Increasing the Quality of Extracted Information by Reading between the Lines. In: Comyn-Wattiau I., du Mouza C., Prat N. (eds.) Ingénierie et management des systèmes d'information—Mélanges en l'honneur de Jacky Akoka

Embley D., Nagy G. (2017) Green Interaction for Extracting Family Information from OCR'd Books (submitted for publication)

Embley D., Zitzelberger A. (2010) Theoretical Foundations for Enabling a Web of Knowledge. In: Proceedings of the Sixth International Symposium on Foundations of Information and Knowledge Systems (FoIKS'10). Sophia, Bulgaria, pp. 211–229

Etzioni O. (2011) Open Information Extraction: The Second Generation. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11). Barcelona, Catalonia, Spain, pp. 3–10

Fader A., Soderland S., Etzioni O. (2011) Identifying Relations for Open Information Extraction. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. Edinburgh, United Kingdom, pp. 1535–1545

FamilySearch. http://familysearch.org

FamilySearch Company Facts. https://media.-familysearch.org/company-facts/

Finkel J., Grenager T., Manning C. (2005) Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In: Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363–370

Fliedl G., Kop C., Mayr H. C. (2003) From Scenarios to KCPM Dynamic Schemas: Aspects of Automatic Mapping. In: Natural Language Processing and Information Systems, 8th International Conference on Applications of Natural Language to Information Systems, June 2003, Burg (Spreewald), Germany, pp. 91–105

Fliedl G., Kop C., Mayr H. C. (2005) From Textual Scenarios to a Conceptual Schema. In: Data and Knowledge Engineering 55(1), pp. 20–37

Fliedl G., Kop C., Mayr H. C., Salbrechter A., Vöhringer J., Weber G., Winkler C. (2007) Deriving Static and Dynamic Concepts from Software Requirements Using Sophisticated Tagging. In: Data and Knowledge Engineering 61(3), pp. 433–448

Fliedl G., Kop C., Mayr H. C., Winkler C., Weber G., Salbrechter A. (2004) Semantic Tagging and Chunk-Parsing in Dynamic Modeling. In: Natural Language Processing and Information Systems, 9th International Conference on Applications of Natural Languages to Information Systems, NLDB 2004, Salford, UK, June 23-25, 2004, Proceedings, pp. 421–426

Gallaire H., Minker J. (eds.) Logic and Data Bases, Symposium on Logic and Data Bases. Advances in Data Base Theory. Plenum Press, New York

Gedcom X. http://www.gedcomx.org/

Grant F. (1912) Index to The Register of Marriages and Baptisms in the PARISH OF KILBARCHAN, 1649–1772. J. Skinner & Company, LTD, Edinburgh, Scotland

Grishman R. (2015) Information Extraction. In: IEEE Intelligent Systems 30, pp. 8–15

Grosz B., Joshi A., Weinstein S. (1995) Centering: A Framework for Modeling the Local Coherence of Discourse. In: Computational Linguistics 21(2), pp. 203–225

Gruber T. (1993) A Translation Approach to Portable Ontology Specifications. In: Knowledge Acquisition 5(2), pp. 199–220

Guizzardi G., Halpin T. A. (2008) Ontological Foundations for Conceptual Modelling. In: Applied Ontology 3(1-2), pp. 1–12

Halpin T. (2004) Business Rule Verbalization. In: Proceedings of the 3rd International Conference on Information Systems Technology and its Applications. Salt Lake City, Utah, pp. 39–52

Halpin T. A., Curland M. (2006) Automated Verbalization for ORM 2. In: On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops, OTM Confederated International Workshops and Posters, AWeSOMe, CAMS, COMINF, IS, KSinBIT, MIOS-CIAO, MONET, OnToContent, ORM, PerSys, OTM Academy Doctoral Consortium, RDDS, SWWS, and SeBGIS 2006, Montpellier, France, October 29 - November 3, 2006. Proceedings, Part II, pp. 1181–1190

Harwood W. (1911) A Genealogical History of the Harwood Families, Descended from Andrew Harwood, Third. Published by Watson H. Harwood, M.D., Chasm Falls, New York

Jiménez P., Corchuelo R., Sleiman H. (2016) ARIEX: Automated Ranking of Information Extractors. In: Knowledge-Based Systems 93, pp. 84–108

Kang H., Getoor L., Shneiderman B., Bilgic M., Licamele L. (2008) Interactive Entity Resolution in Relational Data: A Visual Analytic Tool and Its Evaluation. In: IEEE Transactions on Visualization and Computer Graphics 14(5)

Kim T. (2017) A Green Form-Based Information Extraction System for Historical Documents. MA thesis, Brigham Young University, Provo, Utah

Kop C., Mayr H. C., Zavinska T. (2004) Using KCPM for Defining and Integrating Domain Ontologies. In: Web Information Systems - WISE 2004 Workshops: WISE 2004 International Workshops, Brisbane, Australia, November 22-24, 2004. Proceedings, pp. 190–200

Kushmerick N., Weld D., Doorenbos R. (1997) Wrapper Induction for Information Extraction. In: Proceedings of the 1997 International Joint Conference on Artificial Intelligence, pp. 729–735

Laender A., Ribeiro-Neto B., da Silva A., Teixeira J. (2002) A Brief Survey of Web Data Extraction Tools. In: SIGMOD Record 31(2), pp. 84–93

Laird J. (2012) The Soar Cognitive Architecture. The MIT Press, Cambridge, Massachusetts

Lehnert W., Cardie C., Fisher D., McCarthy J., Riloff E., Soderland S. (1994) Evaluating an Information Extraction System. In: Journal of Integrated Computer-Aided Engineering 1(6), pp. 453–472

Lenat D., Guha R. (1989) Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project. Addison-Wesley Longman Publishing Co., Inc., Boston, Massachusetts

Liddle S., Embley D., Woodfield S. (1993) Cardinality Constraints in Semantic Data Models. In: Data & Knowledge Engineering 11(3), pp. 235–270

Liddle S., Embley D., Woodfield S. (1995) Unifying Modeling and Programming Through an Active, Object-Oriented, Model-Equivalent Programming Language. In: Proceedings of the Fourteenth International Conference on Object-Oriented and Entity-Relationship Modeling (OOER'95). Gold Coast, Queensland, Australia, pp. 55–64

Lindes P. (2014) OntoSoar: Using Language to Find Genealogy Facts. MA thesis, Brigham Young University, Provo, Utah

Lindes P., Laird J. (2016) Toward Integrating Cognitive Linguistics and Cognitive Language Processing. In: Proceedings of the International Conference on Cognitive Modeling (ICCM'16). The Pennsylvania State University, University Park, Pennsylvania

Lindes P., Laird J. (2017) Cognitive Modeling Approaches to Language Comprehension using Construction Grammar. In: Proceedings of the AAAI 2017 Spring Symposium on Computational Construction Grammar and Natural Language Understanding. Technical Report SS-17-02, pp. 213–221

Lindes P., Lonsdale D., Embley D. (2015) Ontology-based Information Extraction with a Cognitive Agent. In: Proceedings of the AAAI-15 Special Track on Cognitive Systems. Austin, Texas, pp. 558–564

Lindes P., Mininger A., Kirk J., Laird J. (2017) Grounding Langauge for Interactive Task Learning. In: Proceedings of the First Workshop on Language Grounding for Robotics. Vancouver, Canada, pp. 1–9

Mahdisoltani F., Biega J., Suchanek F. (2015) YAGO3: A Knowledge Base from Multilingual Wikipedias. In: Proceedings of the 7th Biennial Conference on Innovative Data Systems Research (CIDR 2015). Asilomar, California

Marie A., Gal A. (2007) Managing Uncertainty in Schema Matcher Ensembles. In: Scalable Uncertainty Management, First International Conference, SUM 2007, Washington, DC, October 10-12, 2007, Proceedings, pp. 60–73

Mausam, Schmitz M., Bart R., Soderland S., Etzioni O. (2012) Open Language Learning for Information Extraction. In: Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning. Jeju Island, Korea, pp. 523–534

Mayr H. C., Kop C. (1998) Conceptual Predesign—Bridging the Gap between Requirements and Conceptual Design. In: 3rd International Conference on Requirements Engineering (ICRE '98), Putting Requirements Engineering to Practice, April 6-10, 1998, Colorado Springs, Colorado, Proceedings, p. 90

Mel'čuk I. (1988) Dependency Syntax: Theory and Practice. State University of New York Press, Albany, State University Plaza, Albany, New York

Miller Funeral Home Records, 1917 – 1950, Greenville, Ohio. Darke County Ohio Genealogical Society, Greenville, Ohio

Mitchell T., Cohen W., Hruschka E., Talukdar P., Betteridge J., Carlson A., Dalvi B., Gardner M., Kisiel B., Krishnamurthy J., Lao N., Mazaitis K., Mohamed T., Nakashole N., Platanios E., Ritter A., Samadi M., Settles B., Wang R., Wijaya D., Gupta A., Chen X., Saparov A., Greaves M., Welling J. (2015) Never-Ending Learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15). Austin, Texas, pp. 2302–2310

Müller H., Freytag J.-C. (2003) Problems, Methods, and Challenges in Comprehensive Data Cleansing. HUB-IB-164. Humboldt University Berlin

Nadeau D., Sekine S. (2007) A Survey of Named Entity Recognition and Classification. In: Lingvisticae Investigationes: International Journal of Linguistics and Language Resources 30(1), pp. 3–26

Nagy G. (2012) Estimation, Learning, and Adaptation: Systems that Improve with Use. In: Proceedings of the Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition. Hiroshima, Japan, pp. 1–10

Nagy G. (2017) G. Nagy, DDA: Decision Directed Adaptation. personal communication

Noy N. (2004) Semantic Integration: A Survey of Ontology-Based Approaches. In: SIGMOD Record 33(4), pp. 65–70

Packer T. (2014) Scalable Detection and Extraction of Data in Lists in OCRed Text for Ontology Population Using Semi-Supervised and Unsupervised Active Wrapper Induction. PhD thesis, Brigham Young University, Provo, Utah

Park J. (2015) FROntIER: A Framework for Extracting and Organizing Biographical Facts in Historical Documents. MA thesis, Brigham Young University, Provo, Utah

Rahm E., Bernstein P. (2001) A Survey of Approaches to Automatic Schema Matching. In: The VLDB Journal 10, pp. 334–350

Rahm E., Do H. (2000) Data Cleaning: Problems and Current Approaches. In: IEEE Data Engineering Bulletin 23(4), pp. 3–13

Recasens M., Hovy E. (2009) A Deeper Look into Features for Coreference Resolution. In: Lalitha Devi S., Branco A., Mitkov R. (eds.) Anaphora Processing and Applications, DAARC 2009. Lecture Notes in Computer Science Vol. 5847. Springer, Berlin, Germany

Salton G. (1968) Automatic Information Organization and Retrieval. McGrawHill

Sarawagi S. (2008) Information Extraction. In: Foundations and Trends in Databases 1(3), pp. 261–377

Schone P., Gehring J. (2016) Genealogical Indexing of Obituaries Using Automatic Processes. In: Proceedings of the Family History Technology Workshop (FHTW'16). https://fhtw.byu.edu/archive/2016. Provo, Utah, USA

Settles B. (2010) Active Learning Literature Survey. Computer Sciences Technical Report 1648. University of Wisconsin–Madison

Settles B. (2012) Active Learning. In: Synthesis Lectures on Artificial Intelligence and Machine Learning 6(1), pp. 1–114

Shaffer H. (1997) Shaver/Shafer and Dougherty/Daughery Families also Kiser, Snider and Cottrell, Ferrell, Hively and Lowe Families. Gateway Press, Inc., Baltimore, Maryland

Sleator D. D., Temperley D. (1995) Parsing English with a Link Grammar. In: The Computing Research Repository (CoRR) abs/cmp-lg/9508004

Stroup H. (n.d.) Butler County, Ohio, Cemetery Records Vol. VIII. Hazel Stroup, Hamilton, Ohio

Suchanek F., Kasneci G., Weikum G. (2007) Yago: A Core of Semantic Knowledge. In: Proceedings of the 16th International World Wide Web Conference (WWW 2007). Banff, Alberta, Canada, pp. 697–706

Tao C., Embley D., Liddle S. (2009) FOCIH: Form-based Ontology Creation and Information Harvesting. In: Proceedings of the 28th International Conference on Conceptual Modeling (ER2009). Gramado, Brazil, pp. 346–359

Tijerino Y., Embley D., Lonsdale D., Ding Y., Nagy G. (2005) Toward Ontology Generation from Tables. In: World Wide Web: Internet and Web Information Systems 8(3), pp. 261–285

Turmo J., Ageno A., Català N. (2006) Adaptive Information Extraction. In: ACM Computing Surveys 38(2), pp. 1–47

Vanderpoel G. (1902) The Ely Ancestry: Lineage of RICHARD ELY of Plymouth, England. The Calumet Press, New York, New York

Wong W., Liu W., Bennamoun M. (2012) Ontology Learning from Text: A Look Back and into the Future. In: ACM Computing Surveys 44(4), 20:1–20:36

Woodfield S., Lonsdale D., Liddle S., Kim T., Embley D., Almquist C. (2016) Pragmatic Quality Assessment for Automatically Extracted Data. In: Proceedings of ER 2016 Vol. LNCS 9974. Gifu, Japan, pp. 212–220

Xu L. (2003) Source Discovery and Schema Mapping for Data Integration. PhD dissertation, Brigham Young University, Provo, Utah

Zitzelberger A., Embley D., Liddle S., Scott D. (2015) HyKSS: Hybrid Keyword and Semantic Search. In: Journal on Data Semantics 4(4), pp. 213–299

# Towards FCA-facilitated Ontology-supported Recruitment Systems

Hui Ma[*,a], Sven Hartmann[b], Panrawee Vechsamutvaree[a]

[a] Victoria University of Wellington, New Zealand
[b] Clausthal University of Technology, Germany

Abstract. *Human resources (HR) recruitment is still a major challenge for many organizations since HR recruitment officers need to spend lots of time and effort to find the best candidate from a large number of applicants for a job position. In this paper, we propose a new approach for ontology-supported web-based HR recruitment systems. Our approach is facilitated by Formal Concept Analysis (FCA) for constructing domain-specific ontologies to model position requirements and applicants' competences. To evaluate our approach, we implement a prototype and conduct a case study. The case study demonstrates that the proposed approach has the potential to improve the effectiveness and efficiency of the HR recruitment process.*

Keywords. Ontology • FCA • Human Resources Management

## 1 Introduction

The human resources (HR) recruitment process is often time consuming and involves high operational costs because domain experts are often required to be involved to make decisions (Carroll et al. 1999). Carroll et al. (1999) states that the number of applicants can be up to 1000 to 2000 for one or two vacancies. In this case, HR officers and domain experts have to spend tremendous amounts of time on finding the best candidate for a position from the long list of applicants. To facilitate the access of HR managers and job applicants web-based recruitment systems are used to advertise job positions and collect job applications. After job applications being collected, HR officers and domain experts have to manually go through all the application to select candidates with competencies that best match the job position requirements. One example is tutor recruiting at the School of Engineering and Computer Science (ECS) at Victoria University of Wellington (VUW). In each trimester, a senior tutor (serve as a HR recruitment officer) has to spend a couple of weeks time to manually check the candidate competencies and requirements of tutors and select tutors from a large number of applicants. This process is not only time consuming but also not effective, i. e. may not be able to look at all the possible candidates for each course. Also, domain knowledge about competence and courses are required from course lecturers. It is important to capture the domain requirements so that quality solutions can be produced (Kop and Mayr 2008).

To improve efficiency and performance of the recruitment process, the use of a domain-specific ontology has been proposed for recruitment systems and competency management (Draganidis et al. 2006; Fazel-Zarandi and Yu 2008). Ontologies have been used for skill management, expertise finding and competency management (Woelk 2002), The benefits of ontology-supported

* Corresponding author.
E-mail. hui.ma@ecs.vuw.ac.nz

recruitment systems are reported in (Baader et al. 2003; Bizer et al. 2005; Maedche and Staab 2001; Mochol et al. 2006; Trichet and Leclere 2003; Woelk 2002; Zhang 2007).

An ontology identifies the relevant concepts and relationships among the concepts that exist in a domain (Gruber 1995). A well designed ontology can provide good support for machine processing and be understandable for human users (Baader et al. 2003; Bachmann et al. 2007). Building an ontology is often time consuming and requires the involvement of many domain experts. Also, different organizations have different business contexts and need organization-specific ontologies for their system. Many organizations, however, face the challenge that there is neither an existing ontology to be used nor sufficient expertise available to build an ontology. Formal Concept Analysis (FCA) has been proposed for building ontologies without the involvement of domain experts. Hwang et al. (2005) uses FCA to express an ontology in a lattice. The lattice is easy to understand for human users and can serve as a basis for ontology building. However, there is a lack of research on the use of FCA for building ontologies for HR recruitment systems.

The aim of this paper is to propose an ontology-supported approach for web-based recruitment systems which exploits FCA for building ontologies so that the HR recruitment process can be improved. In particular, we will present an ontology design approach that uses FCA to construct a domain-specific ontology. The constructed ontology is then used to automatically match applicants to job vacancies to reduce the involvement of domain experts. To demonstrate the effectiveness and efficiency of our approach, we develop a prototype system and conduct a case study for tutor matching.

This paper is organized as follows. In Section 2 we give a motivating example. Section 3 provides preliminaries that we will need later on. In Section 4 we present our proposed approach for ontology-supported HR recruitment where FCA is used for building domain-specific ontology. Section 5 reports on a case study that we have conducted to gain experience with the proposed approach. Section 6 concludes the paper and makes suggestions for future work.

## 2 A Motivating Example

HR recruitment often involves the processing of a large number of applications for a job vacancy. The School of Engineering and Computer Science (ECS) at VUW recruits about 150 tutor positions each trimester for about 50 courses at different levels.

When students want to apply for a tutor position at the school, they have to use an online portal where they provide personal information, such as their student ID, name, email, phone, degree completed, degree enrolled, year of study, major, tutoring experience, and preferred courses. Further information about the courses that the applicants have completed at VUW can be retrieved from the course results database using the applicant's student ID.

Different courses require different competences for their tutors. Based on the requirements for a particular course, a senior tutor manually selects a list of candidates which is then provided to the respective coordinator of the course to make the final decision. The matching of tutors to courses is based on a combination of several competence factors. For example, an ideal tutor of a course is someone who has the experience of tutoring the same course before. If she/he has not tutored the course before, then she/he should have studied the course at VUW before with a good grade, say grade B or better. Moreover, the education level of the candidate should not be lower than the course level.

If no candidate satisfies all the requirements of a course, then the senior tutor and the course coordinator need to work together to find someone who matches most of the requirements. For example, a candidate can be considered for a course if she/he has not done the course at VUW before, but has done some post-course of the course. To measure the degree of matching of the requirements, ranking methods can be used. In the

literature, the matching between job positions and job candidates could be done by measuring the distance between respective concepts in an ontology hierarchy. One can use a standard competence dictionary such as the IEEE Reusable Competency Definition (IEEE RCD 2008) or DISCO (Müller-Riedlhuber 2009). However, for this case there is no exiting domain-specific ontology that can be used.

## 3 Preliminaries

In this section we will recall relevant background information about ontologies and FCA.

### 3.1 Description Logics

Description logics (DL) are a family of logic languages for representing knowledge and reasoning about it (Baader et al. 2003). The knowledge of interest for some application is represented in terms of individuals, concepts and roles, and is stored in a knowledge base. Concepts stand for sets of individuals, while roles stand for relationships to other individuals. A *DL knowledge base* is a set of axioms, and usually consists of two parts: a terminological knowledge layer (called TBox) and an assertional knowledge layer (called ABox). The TBox describes the terminology in use for the application, that is, defines the concepts and captures additional constraints on their interpretation. The ABox describes the individuals, that is, contains assertions that relate individuals to concepts.

We briefly review syntax and semantics of description logics. Let $N_C$ and $N_R$ be fixed sets of concept names and of roles names, respectively. One can then build complex concept expressions out of them by using the concept constructors provided by the particular description logic being used (see Looser et al. 2013). Let $C$ denote the set of complex concept expressions that can be obtained by finitely many applications of these constructors. The members of $N_C$ are called atomic concepts. We further use the empty concept $\bot$ as a shortcut for $\neg\top$, and $(\leq m).R$ as a shortcut for $\neg(\geq m+1).R$.

Different description logics vary by the concept constructors that they permit. The choice of a particular description logic is usually done by balancing expressiveness against the complexity of the associated decision problems. In this paper, we use $\mathcal{ALN}$ which is among the most popular description logics (Liao et al. 1999). However, the ideas discussed in this paper are general in the sense that they can be easily tailored to other reasonably expressive description logics. Note that $\mathcal{ALN}$ is included in $\mathcal{SHOIN}^{(\mathcal{D})}$, the description logic underlying OWL-DL.

A *subsumption axiom* is a statement of the form $C_1 \sqsubseteq C_2$ with concepts $C_1, C_2$ in $C$. A *terminology* (or TBox) $\mathcal{T}$ is a finite set of subsumption axioms. We use the shortcut $C_1 \equiv C_2$ to denote both $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$, and call $C_1, C_2$ *equivalent*.

Interpretations are used to assign meaning to syntactic constructs. An interpretation $\mathcal{I}$ consists of a non-empty interpretation domain $O$ and an interpretation function $\mathcal{I}(.)$, which assigns to each atomic concept a subset of $O$, and to each role a binary relation on $O$. The interpretation $\mathcal{I}$ can be easily extended to concept expressions in $N_C$. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{T}$ if $\mathcal{I}(C_1) \subseteq \mathcal{I}(C_2)$ holds for every subsumption axiom $C_1 \sqsubseteq C_2$ in $\mathcal{T}$. A model is finite if the interpretation domain $O$ is finite. In this case, the model is also said to be an *instance* (or ABox) of $\mathcal{T}$.

A concept $C_1$ *subsumes* a concept $C_2$ if $\mathcal{I}(C_1) \subseteq \mathcal{I}(C_2)$ holds for every instance $\mathcal{I}$ of $\mathcal{T}$. We also write $\mathcal{T} \models C_1 \sqsubseteq C_2$. We use $\mathcal{T} \models C_1 \equiv C_2$ as a shortcut for $\mathcal{T} \models C_1 \sqsubseteq C_2$ and $\mathcal{T} \models C_2 \sqsubseteq C_1$.

**Example 1** For our HR recruitment system we may use the subsumption axiom $\texttt{SWEN301} \sqsubseteq \exists\texttt{hasTask.EssayMarking}$, $\texttt{SWEN301} \sqsubseteq \exists\texttt{isPostcourse.SWEN222}$ to state that course SWEN301 has a task of essay marking and is a post-course of SWEN222.

The subsumption problem asks whether $\mathcal{T} \models C_1 \sqsubseteq C_2$ holds for concepts $C_1, C_2$ and a TBox $\mathcal{T}$. The satisfiability problem asks whether $\mathcal{T} \models C \equiv \bot$ holds for a concept $C$ and a TBox $\mathcal{T}$. Both problems are decidable for description logics such

as $\mathcal{ALN}$. For details, we refer to Baader et al. 2003.

### 3.2 Formal Concept Analysis

It has been shown that FCA can help to structure and build an ontology for particular application of interest (Cimiano et al. 2005; Hwang et al. 2005). FCA provides a formal framework for recognizing groups of elements that exhibit common properties. It is a theory of data analysis which identifies conceptual structures among data sets. These structures can be graphically represented as conceptual lattices which allow the analysis of complex structures and the discovery of dependencies within the data. FCA can be used for visualizing the ontology in the form of a lattice, in order to support navigation and analysis tasks. Representing an ontology as a lattice makes it easy to understand for human users.

Next, we review basic notions from FCA (Ganter and Wille 1999). A *formal context* is a triple $A \times B\ K := (O, C, R)$, where $O$ is a set of individuals, $C$ is a set of properties (or attributes), and $R$ is a relation $R \subseteq O \times C$ that links each individual $o \in O$ to the properties satisfied by $o$. That is, $(o, b) \in O$ states that $O$ has property $b$. A *formal concept* (or FCA concept) is a pair $(A, B)$ such that $A$ and $B$ are maximal with $A \times B \subseteq I$. The set $A \subseteq O$ is called the *extent*, while the set $B \subseteq C$ is called the *intent* of the concept. A formal concept $(A_1, B_1)$ is a *subconcept* of a formal concept $(A_2, B_2)$ if $A_1 \subseteq A_2$ (or equivalently $B_1 \supseteq B_2$). The concept lattice $\mathfrak{B}(\mathbb{K})$ of $\mathbb{K}$ is the set of all its FCA concepts together with the subconcept/superconcept relation.

As usual, we use lattice diagrams to illustrate the ordering relation among the formal concepts in a concept lattice. The nodes of the lattice are labelled by the respective FCA concepts. For an example, see Figure 4. For the sake of clarity, however, node labels do not show the full extent and intent of an FCA concept, but show an individual $o$ only in the most specific FCA concept it belongs to.

Contexts can be represented as tables, with columns headed by the properties $b$ and rows headed by individuals $a$. The cells of the table are marked if and only if the relations holds for the corresponding pair of individual and property. For an example see Figure 3.

## 4 Ontolgy-supported HR Recruitment using FCA

In this section, we present our approach for an ontology-supported web-based HR recruitment system where FCA is used to build a domain-specific ontology.

### 4.1 Framework of an Ontology-supported Web-based HR Recruitment System

Figure 1 shows our framework of an ontology-supported web-based HR recruitment system.

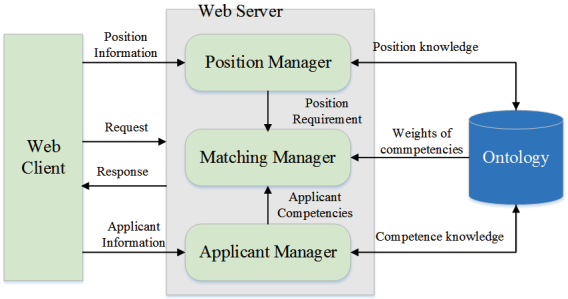



*Figure 1: Framework of a web-based HR recruitment system*

In the front-end, we have Web Client which connects to Web Server using HTTP request and HTTP response. In the Web Server, there are three modules which are Position Manager, Applicant Manager and Matching Manager. The three modules in Web Server connect to the Ontology for storing and retrieving the data. Position Manager is used for managing job positions, such as adding a new position or editing position requirements. Applicant Manager is used for managing applicants' profiles, such as adding a new applicant, editing the profile of an applicant, or searching applicants. Matching Manager is used for managing the position-candidate matching. For matching applicants to positions, the system compares the position requirements retrieved by Position Manager and the applicant's competencies retrieved

by Applicant Manager. Matching Manager is also used for managing the weights of each competency.

Often, applicants do not match all the requirements. To rank candidates for a position, we need a ranking method. We will assign weights to relevant competence factors. The matching of different competencies have different weights since the importance of each competency to a position is different. For example, for tutor-course matching, tutoring experience is weighted higher than the education level of a candidate.

## 4.2 FCA-based Ontology Building

To collect the domain knowledge that can be used for selecting candidates we propose to use FCA. Figure 2 shows the steps of our FCA-based method of building domain-specific ontologies from candidate application information. Note that one can take advantage from candidate application instances collected from previous years to build the ontologies.

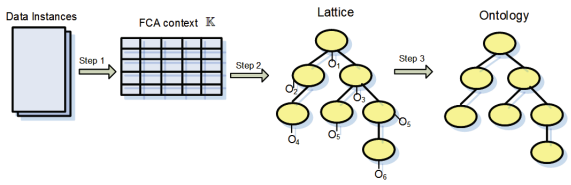



*Figure 2: FCA-based ontology building process*

We first create a formal context matrix. Then, we use this formal context matrix to build the concept lattice for an ontology. In one organization, we many need to build several contexts. For example, for the tutor recruitment system, we use FCA to build a prerequisite course context and an education level context.

Figure 3 shows an example of a context $K(O, C, R)$ of course prerequisites. The set of objects $O$ is a collection of candidates, denoted by Alex, Bob, etc. The set of attributes $C$ is the set of courses. If a candidate has already passed a course, then the relationship $R$ holds and we mark it by "X" in the table. The concept lattice in Figure 4 is derived from the context in Figure 3.

Each node in this lattice, illustrated by a circle, is a formal concept. For example, one of the formal concepts of the context described in Figure 3 is {Alex, Eric} × {COMP103, SWEN223}, where the set {Alex, Eric} is the extent of the concept, while the set {COMP103, SWEN223} is its intent. According to Figure 4, the formal concept {Alex, Bob} × {COMP103, SWEN221, SWEN222} is a sub-concept of the concept {Alex, Bob, Candy} × {COMP103, SWEN221}.

| | SWEN222 | SWEN223 | SWEN301 | SWEN221 | COMP103 | SWEN423 |
|---|---|---|---|---|---|---|
| Alex | X | X | X | X | X | |
| Bob | X | | | X | X | |
| Candy | | | | X | X | |
| Daniel | | | | | X | |
| Eric | | X | | | X | |
| Flona | X | X | X | X | X | X |

*Figure 3: Formal context of course pre-requisites*

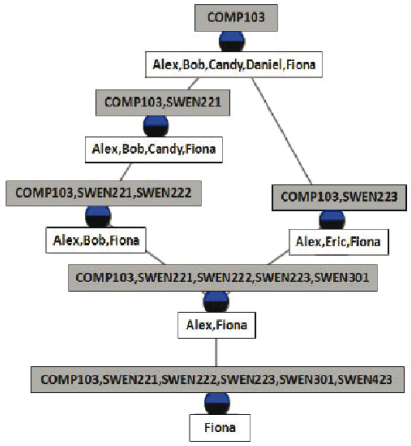



*Figure 4: Concept lattice of course pre-requisites*

To build an ontology, we can use the reduction procedure proposed in (Haav 2004). The reduction procedure has two steps. The first step is to eliminate the redundant elements. For example, a pair $(A, B)$ and $B$ (intent) appear in every descendant. We can eliminate the inherited elements. The second step is to eliminate the lattice of extents. Figure shows an example of a resulting ontology.

## 5 A Case Study

To demonstrate the effectiveness and efficiency of our proposed approach, we have implemented a

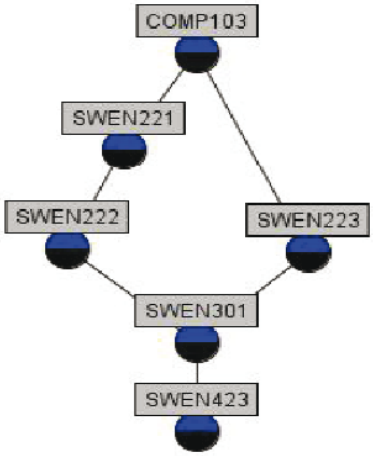



*Figure 5: Ontology of course pre-requisites*

prototype of a web-based tutor recruitment system using our proposed framework and used the proposed FCA-based ontology building method to build ontologies. We have then designed a test case, based on real data at ECS. We used the prototype to select a shortlist of candidates for each tutor position. In the mean time we invited the senior tutor to select a list of candidates for each course. For this case study the senior tutor was asked to list the top 3 suitable candidates. We have then compared the quality of the results and the time used by the two different approaches.

### 5.1 Prototype of an Ontology-supported Recruitment System

We have developed a prototype to demonstrate the performance of our proposed framework. We used Apache Jena$^{TM}$ to store and manipulate our ontologies, so that SPARQL could be used for querying. We used our FCA-based approach to build ontologies needed by the tutor recruitment system. We collected information on tutor applicants and courses, and constructed ontologies to capture the relationships between courses, degrees, and course requirements.

For each course, the prototype system retrieves the course requirements and the applicants' competency information from the ontologies. Then the Matching manager compares the course requirements to the applicants' competencies and calculates the fitness value of each qualified applicant, and displays a shortlist with the required number of ranked candidates, starting with the highest fitness. Often, the candidates do not match all the requirements. The fitness value permitted the ranking of candidates.

In this case study, we considered the fitness of a candidate as a weighted sum of different competencies, e. g., education level, experience:

$$Fitness = w_{EC} + w_L + w_{EP} + w_C$$

Herein, $w_{EC}, w_L, w_{EP}, w_C$ denote the weights assigned for having experience of tutoring the course, education level, having experience of tutoring a post-course, having done the course. We have normalized the weights.

For each course, the weights of competencies are different. For example, for matching tutors to courses, experiences of being a tutor of the course has higher weight than the education level of the candidate. In this way, the system ranks a candidate who has not completed a Bachelor degree but has tutored this particular course before higher than a candidate who has a Master degree but has not tutored the course before.

### 5.2 Results

In this section we compare the results obtained with our system to the results obtained by the senior tutor using the existing approach. Due to the page limit we only discuss some example courses in this article.

Firstly, we compare the time used by the two approaches. Recall that there are 150 applicants a tutor position per trimester. Using the current approach the senior tutor needs to retrieve the information of the candidates and courses from different resources and then manually selects suitable candidates by using her domain knowledge of courses or by consulting the respective course lecturers. Using the current manual system the senior tutor needs to spend 75 hours in total to check first tutors for each courses, assuming 30 minutes for each candidate. Using our prototype she only needs to spend a few minutes.

Secondly, we compare the lists selected by the senior tutor and the ones produced by our prototype, see Table 1 and Table 2, respectively. For most courses, the results obtained by the two different approaches are very similar, except for COMP307. One candidates has been selected by the senior tutor because this candidate got good grades in other courses. However he had neither studied nor tutored COMP307 and its post-courses before. Candidates who studied COMP307 before got grades lower than B and, hence, are not selected. On the other hand, our system listed 300198173 and 300198193 as the most suitable candidates since they both got A+ for COMP422 which is a post-course of COMP307. The results obtained by our system are better because the two candidates achieved good grades in COMP422, and thus should have knowledge of COMP307 which is a prerequisite of COMP307. From these results, we can see that without using our system, potential candidates can be missed out.

| | 1st | 2nd | 3rd |
|---|---|---|---|
| SWEN221 | 300198187 | 300198201 | 300198202 |
| SWEN222 | 300198201 | 300198180 | 300198187 |
| SWEN307 | 300198201 | - | - |
| SWEN304 | 300198175 | 300198176 | 300198182 |

*Table 1: Matching results from the senior tutor*

| | 1st | 2nd | 3rd |
|---|---|---|---|
| SWEN221 | 300198201 | 300198187 | 300198202 |
| SWEN222 | 300198201 | 300198180 | 300198187 |
| SWEN307 | 300198173 | 300198176 | 300198178 |
| SWEN304 | 300198176 | 300198182 | 300198175 |

*Table 2: Matching results from our prototype*

From the above results, we can see that our prototype has the potential to find suitable candidates due to the usage of ontologies that model the relationship between courses. Some of the candidates might be missed out using the existing approach. The propotyope system has huge potential to assist the senior tutor by reducing the time and effort needed for matching the applicants to courses.

## 6 Conclusion

In this paper, we have proposed an approach of ontology-supported web-based HR recruitment that exploits FCA for building domain-specific ontologies. We developed a prototype of our proposed approach and conducted a case study for tutor-course matching at ECS of VUW. The case study shows that our proposed approach has indeed the potential to improve the efficiency and effectiveness of the HR recruitment process.

## References

Baader F., Calvanese D., McGuinness D. L., Nardi D., Patel-Schneider P. F. (eds.) The description logic handbook: theory, implementation, and applications. Cambridge University Press

Bachmann A., Hesse W., Ruß A., Kop C., Mayr H. C., Vöhringer J. (2007) A Practical Approach to Ontology-based Software Engineering. In: Proceedings of the 2nd International Workshop on Enterprise Modelling and Information Systems Architectures (EMISA'07), pp. 129–142

Bizer C., Heese R., Mochol M., Oldakowski R., Tolksdorf R., Berlin F. U., Eckstein R. (2005) The Impact of Semantic Web Technologies on Job Recruitment Processes. In: In Proceedings of the 7th International Conference Wirtschaftsinformatik, pp. 1367–1383

Carroll M., Marchington M., Earnshaw J., Taylor S. (1999) Recruitment in small firms: Processes, methods and problems. In: Employee Relations 21(3), pp. 236–250

Cimiano P., Hotho A., Staab S. (2005) Learning Concept Hierarchies from Text Corpora Using Formal Concept Analysis. In: Journal of Artificial Intelligence research 24, pp. 305–339

Draganidis F., Chamopoulou P., Mentzas G. (2006) An ontology based tool for competency management and learning paths. In: 6th International Conference on Knowledge Management (I-KNOW 06), pp. 1–10

Fazel-Zarandi M., Yu E. (2008) Ontology-Based Expertise Finding In: Practical Aspects of Knowledge Management: 7th International Conference (PAKM 2008), Proceedings Yamaguchi T. (ed.) Springer, pp. 232–243

Ganter B., Wille R. (1999) Formal Concept Analysis. Springer

Gruber T. R. (1995) Toward principles for the design of ontologies used for knowledge sharing? In: International journal of human-computer studies 43(5-6), pp. 907–928

Haav H.-M. (2004) A Semi-automatic Method to Ontology Design by Using FCA. In: Proceedings of the CLA 2004 International Workshop on Concept Lattices and their Applications

Hwang S.-H., Kim H.-G., Yang H.-S. (2005) A FCA-based ontology construction for the design of class hierarchy. In: Computational Science and Its Applications–ICCSA 2005, pp. 307–320

IEEE RCD (2008) IEEE Standard for Learning Technology - Data Model for Reusable Competency Definitions

Kop C., Mayr H. C. (2008) Templates in Domain Modeling - A Survey. In: The Evolution of Conceptual Modeling - From a Historical Perspective towards the Future of Conceptual Modeling [outcome of a Dagstuhl seminar held 2008]., pp. 21–41

Liao M., Hinkelmann K., Abecker A., Sintek M. (1999) A competence knowledge base system as part of the organizational memory. In: German Conference on Knowledge-Based Systems. Springer, pp. 125–137

Looser D., Ma H., Schewe K. (2013) Using Formal Concept Analysis for Ontology Maintenance in Human Resource Recruitment. In: Ninth Asia-Pacific Conference on Conceptual Modelling (APCCM), pp. 61–68

Maedche A., Staab S. (2001) Ontology learning for the semantic web. In: IEEE Intelligent systems 16(2), pp. 72–79

Mochol M., Nixon L. J. B., Wache H. (2006) Improving the Recruitment Process through Ontology-based Querying.. In: SEBIZ. CEUR Workshop Proceedings Vol. 226

Müller-Riedlhuber H. (2009) The European Dictionary of Skills and Competences (DISCO): an example of usage scenarios for ontologies. In: Proc. 5th International Conference on Semantic Systems, ISEMANTICS. JUCS Conference Proceedings Series, pp. 467–479

Trichet F., Leclere M. (2003) A Framework for Building Competency-Based Systems Dedicated to Human Resource Management In: Foundations of Intelligent Systems: 14th International Symposium (ISMIS 2003), Proceedings Springer, pp. 633–639

Woelk D. (2002) E-Learning, Semantic Web Services and Competency Ontologies. In: Proceedings of EdMedia: World Conference on Educational Media and Technology 2002. Association for the Advancement of Computing in Education (AACE), pp. 2077–2078

Zhang J. (2007) Ontology and the semantic web. In: Ontology and the Semantic Web 2007. dLIST, pp. 9–20

# Towards Effectiveness Assessment of Domain Modelling Methods and Tools in SPL Development

Mykola Tkachuk[*,a,b], Rustam Gamzayev[a], Iryna Martinkus[a], Volodymyr Sokol[a], Oleh Tovstokorenko[a]

[a] Department of Software Engineering and Management Information Technologies, National Technical University "Kharkiv Polytechnic Institute", Ukraine
[b] Department of Systems and Technologies Modelling, V.N. Karazin Kharkiv National University, Ukraine

Abstract. *Domain-driven design (DDD) and especially the usage of domain modelling methods (DMM) are modern approaches to improve software quality, and a way to develop software product lines (SPL). To emphasize advantages of DDD and DMM usage, a 3-level design scheme is proposed, which reflects also variability issues in the framework. According to this metaphor the main attention is paid to the phases of logical domain modelling and physical modelling, with usage of two alternative DMM-methods: JODA and ODM approaches. The algorithmic model for an efficiency coefficient estimation of alternative DMM usage is proposed, which utilizes structured data resources, expert methods and metrics used in SPL development processes. A feasibility study for the proposed approach is provided and the obtained experimental results are discussed, which allow to make positive conclusions about this research and to outline its further steps to be done.*

Keywords. Domain Modelling Methods • Software Product Line • Variability • Structural Complexity • Efficiency Coefficient • Code Reusing • Metric

## 1 Introduction: Research Actuality and Goals

The largest amount of existing methodologies for software development, and in the first place modern agile-methods are supposed to achieve the following two main goals proposed by Sommerville (2011):

1. to decrease the project's costs with respect to all specified functional requirements and quality attributes to be implemented in a target software system;
2. to reduce the time needed for delivering of this software product on consumer market.

One of the most effective way to resolve this problem is reusing different project solutions (assets): domain knowledge, requirements specifications, software architectures, design patterns, and finally source (program) code. This approach is the basis of advanced concepts of software engineering such as the development of software products lines and software factories (Greenfield and Short 2004), as well as methods of software variability management (Capilla et al. 2013). In turn, in order to achieve an appropriate level of assets reuse in these processes, the methodology of domain-driven design (DDD) is widely used (Evans 2004; Tune and Millet 2015), where a concept of domain model (DM) as a core for conceptualization and reuse of expert knowledge within the given application area (universe of discuss - UoD) was elaborated: e. g. Karagiannis et al. (2016), Michael and Mayr (2015) and others. The

* Corresponding author.
E-mail. tka.mobile@gmail.com

DDD approach is successfully used to develop software system families (SSF) and software product lines (SPL), which is one of the a main trends in modern software engineering (Klaus 2016). It is to mention that these 2 notions sometimes are used as synonyms because they define a specific collection of software components, which have both general and specific functional properties, and include a special mechanism to provide software variability (Capilla et al. 2013) in order to be configured for multiple reuse for solving of different problem-specific tasks in an appropriate UoD.

But some authors, e.g. Kittlaus and Clough (2009) and Lee (2015) define an important difference between SSF and SPL, namely: the members of SSF as usually are used together to solve some tasks in an appropriate UoD, while single items of any SPL can be applied autonomously in particular software applications.

As a example of SSF the collection of service-oriented software solutions on cloud-based platform Oracle SOA Suite 11g (Oracle Corp n.d.) can be considered, and as a typical example of SPL a well-known office software package MS Office (Microsoft Corp n.d.) can be named. In the following some development problems of SPLs are examined, which are defined more correctly in the manuals of the world-wide recognized international organization in the IT-domain, the Software Engineering Institute at the Carnegie Mellon University (Software Product Line Conference n.d.), namely *«... software product line is a set of software-intensive systems that share a common, managed set of features satisfying the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way... ».* Due to these properties, the usage of SPL concepts provide, in particular, the following advantages in comparison with separate software system development, namely:

- less project costs (up to 60%),
- time decreasing for release of a software product on the market (in some cases up to 90%)
- significant reduction of IT-project's staff (up to 70%), and some others.

Taking the issues mentioned above into account, the main objective of the research presented in this article is to propose a sophisticated approach to effectiveness estimation of DMM usage in the SPL development. The remaining article is organized in the following way: Section 2 analyses briefly related work in the SPL domain, provides the classification of existing DMMs and shows the results of the comparative analysis of some CASE-tools used for support. In section 3 the proposed research methodology is outlined, which emphasizes variability issues in the domain modelling conceptual scheme, and includes a collection of heuristic assumptions combined with formalized specifications to define an efficiency coefficient for alternative DMM usage. In section 4 we present the algorithmic model for the estimation of this efficiency coefficient in different project situations, which is based on structuring and analysing of domain-specific knowledge about interconnected and complex data resources, expert methods and metrics. Section 5 presents the first implementation issues and results of the test-cases to prove the proposed approach. In section 6 the article concludes with a short summary and with an outlook on next steps to be done.

## 2 Software Product Lines Development with Domain Modelling: Related Work and Open Problems

### 2.1 Typical Structure and Features of Software Product Lines

Considering typical SPL structures (can be found in Klaus (2016)), usually its software components can be categorized into three main groups (see Figure 1), namely:

i constant components, which form so-called core of SPL;

ii variable components that already exist, and which can be customized for the specific usage with special features as a part of this SPL;

iii new components that have to be developed additionally in this SPL, taking into account new customer's functional requirements etc.

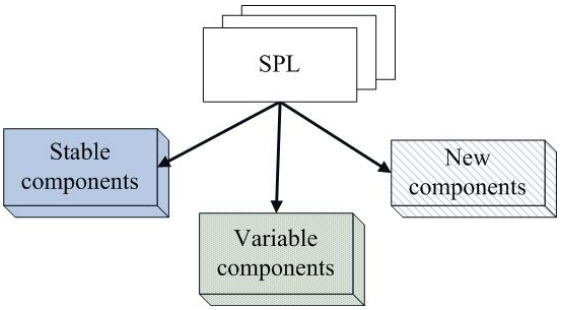



*Figure 1: Reference SPL structure*

Summarizing the research presented in Nagesh and Vivek (2016), components of any SPL can be divided into their three main types (i) – (iii) in the following way (see Figure 1):

- constant components (group i): they are the components, that have a number of functional changes not higher than 25%;
- variable components (group ii): these components have a number of functional changes within range of 25% - 75%;
- new components (group iii): they represent the components, that have a number of functional changes more than 75%.

Main problems of development and efficient SPL implementation are considered by many experts, e. g. in Pohl et al. (2010); Perovich et al. (2009); Bayer et al. (2004), and the relevance of those works proved by a representative international scientific and practical conference: The Software Product Line Conference (n.d.) (SPLC), which has been held annually for more than 20 years. So, in particular, among these problems the following can be identified:

- design and evaluation of SPL reference architectures,
- development of CASE-tools and code frameworks for SPL implementation and maintenance,
- advanced requirements management in SPL development,
- transformation legacy software into new SPL, and some others.

At the same time, the provided analysis of the obtained results in the SPL problem domain shows, that the following important issues still remain insufficiently elaborated, namely:

- building of quantitative metrics for SPL components complexity assessment, which have an impact on the degree of code reuse extent;
- determination of structural and functional complexity of a DM which is further used for code generation in the SPL development;
- elaboration of approaches to effectiveness estimation of alternative DMM usage in order to improve the quality of SPL construction.

These problems are considered more detailed in the following sections.

## 2.2 Classification of Domain Modelling Methods

During the last 10-15 years a lot of different domain modelling methods (DMM) were developed (Ferré 1999; Kelly and Tolvanen 2008). Despite of their differences from the design point of view, the most suitable way to classify DMMs, is to consider them by type of phases / artefacts to be reused in a software development process. Based on this suggestion, the following groups of these methods should be considered (see Figure 2):

1. DMM for requirements reusing;
2. DMM for architectures reusing;
3. DMM for assets reusing;
4. DMM for component reusing.

Taking our main research goal into account: to identify the effectiveness of different DMM usage in the SPL development, we have to consider the methods from two similar groups more detailed: (1) and (3) respectively. Therefore, two appropriate methods, namely JODA and ODM were chosen and briefly presented below. The JODA method (Joint integrated avionics Object oriented Domain Analysis (Ferré 1999)) uses an object -

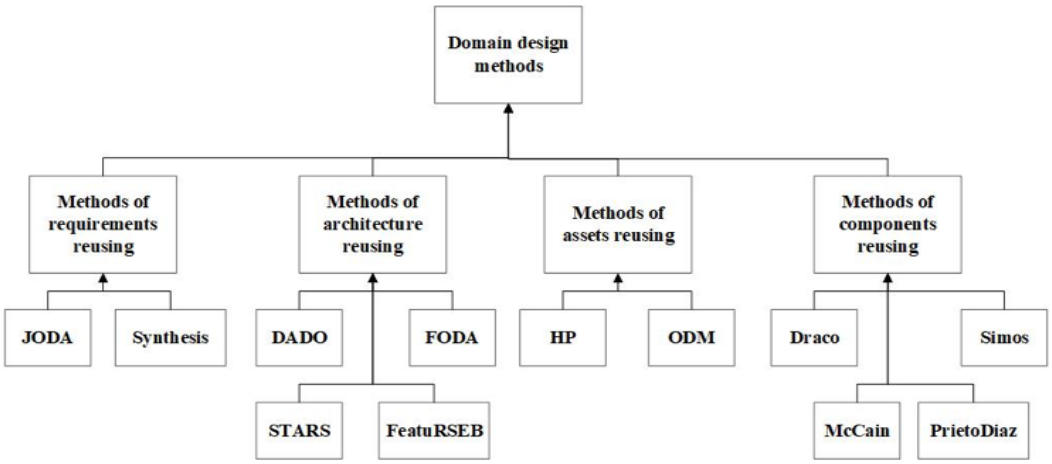



*Figure 2: Taxonomy of domain modelling methods.*

oriented approach to cover the domain analysis phase, and includes the following processes:

- Domain data preparation: identification and gathering of appropriate data sources, references and software artefacts, which are relevant for a given domain.
- Domain scope definition: elaboration of diagrams for higher-level entities, identify of generalization-specialization, aggregation and other relations within domain, build a domain glossary.
- Domain modelling: identification, definition and modelling several domain scenarios in order to group domain-specific objects and activities to represent them in next domain engineering process.

The ODM method (Organizational Domain Modelling (Ferré 1999)) systematically supports the mapping of domain-specific artefacts into reusable assets, that can be reused in future software development activities. This approach includes the following phases:

- Plan domain engineering: this one is focused on understanding stakeholders and defining the domain analysis scope.
- Domain modelling: it concerns collecting and documenting the domain-specific information resources which are relevant for future reusing.
- Domain assets base: the final phase of the ODM method that supposes defining the project scope, creating (choosing) system architectures and implementing a physical asset base for the given domain.

In order to support all main phases / activities with any DA&DSM method the appropriate CASE-tool has to be used, and a short overview of them is given in the next paragraph.

## 2.3 Comparative Analysis of Domain Modelling CASE-tools

Generally, visual modelling tools in Software Engineering have evolved a lot in recent years. One of the new trends in this domain is the transition from unified modelling environments like UML or SysML (OMG 2010), to domain-specific modelling (DSM) languages and tools, e.g. WebML, SoaML, and some others (Reinhartz-Berger 2013). These DSM - approaches allow developers to design and to analyse software in terms of a target problem domain, and finally to generate an application source code in different programming languages based on high-level requirements specifications. It is to mention that existing CASE-tools for DSM are quite varied in their capabilities,

and for comparing them, it is necessary to choose a set of criteria. Obviously there are a lot of different ways to define such criteria configurations, but generic enough and in the same time a practice-oriented one is the following list of criterion: a possibility to generate code by domain model, a possibility to build model by code, and last but not least mandatory license. The appropriate data about wide-used CASE-tools to support DMM are shown in Table 1.

*Table 1: Results by comparative analysis of domain modelling CASE-tools*

| | Generate code by model | Build model by code | Mandatory license |
|---|---|---|---|
| Eclipse Modeling Framework | + | - | + |
| Rational Rose | + | + | + |
| FeatureIDE | + | + | - |
| Visual Paradigm | + | + | ? |
| Actifsource | + | + | - |

The legend here is the follows: "-" means "is not available"; "+" means “is available for free”; and "?" means “a proprietary license is needed”.

Taking into account the data presented in Table 1 as the dedicated CASE-tools for future DM implementation the Actifsource and Eclipse Modeling Framework (EMF) were chosen.

## 3 Research Methodology Outline

### 3.1 Variability Issues in Domain Modelling: A Conceptual Scheme

Nowadays DDD in considered as a recognized methodology to build a complex software in different application areas with respect to this important challenge: to provide a high level of assets reusability in a given project. Although main essential advantages and limitations of DDD are already discussed intensively in many recent publications, from our point of view the positive core of the DDD-methodology can be emphasized once again if we consider an analogy between the DDD-approach in software development and the well-known 3-level vision about data representation in database development (Batini et al. 1992) (see Figure 3).

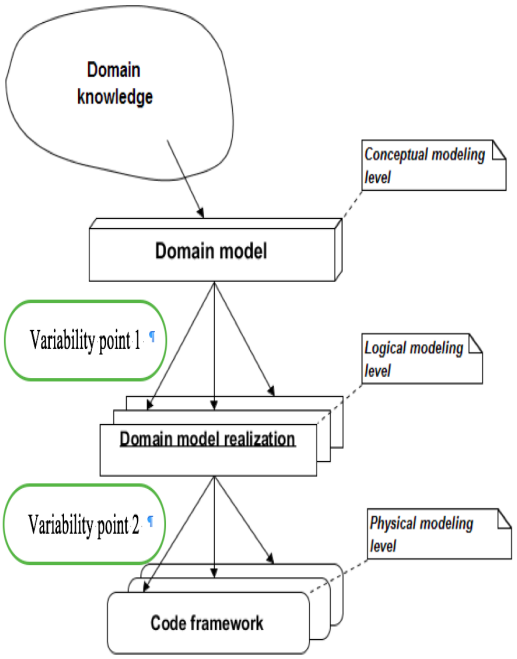



*Figure 3: 3-level scheme in DDD approach.*

Note that, according to this vision about the DDD approach for the one and the same domain model (at conceptual modelling level), a lot of different of its realizations (at the logical modelling level) can be constructed. And for each of them an appropriated code framework might be generated finally (at the physical modelling level) using CASE-tools. In fact, in this scheme we are facing with a variability problem (see in Figure 3 "Variability point 1" and "Variability point 2"), which is now one of the main challenges in modern software development (Capilla et al. 2013).

### 3.2 An Approach to Effectiveness Estimation of Domain Modelling Methods (DMMs): Heuristic Rules and Formalization

In order to evaluate an influence of different factors on effectiveness of DMM usage within a SPL development framework, it is necessary to take a

large number of complicated and poorly formalized information processes and objects into account. This is the reason why we propose to develop an appropriate algorithmic model for this purpose (Patdar 2014). The methodological basis for constructing this model is the following system of five heuristic assumptions, which were formulated based on the study of modern information sources and the generalization of our own experience in software development using the methods of domain modelling, namely:

*Assumption 1.* There exists a certain set of domain modelling methods (set M):

$$(DMM)_i \quad \epsilon \quad M, i = 1, 2, 3... \tag{1}$$

where DMM (DomainModelingMethod) is an identifier denoting a separate method, that can be used to construct an appropriate DM. There also exists the set of relevant technologies T:

$$(DMT)_j \quad \epsilon \quad T, j = 1, 2, 3... \tag{2}$$

where DMT (DomainModelingTechnology) is an identifier denoting a particular technology (with a CASE-tool) for implementing the appropriate DMM.

As a result of the application of a particular DMM with the corresponding DMT, the target DM from the set D can be constructed for the given UoD:

$$(DM)_{i,j} \quad \epsilon \quad D, \tag{3}$$

where $(DM)_{i,j}$ is a domain model obtained as a result of the application of the i-th DMM and j-th DMT respectively.

*Assumption 2.* All DM from the set D have different complexity levels, so there is such a mapping ρ:

$$p : D \rightarrow DMC, \tag{4}$$

where DMC (DomainModelComplexity) is a set of possible values of the quantitative level of structural and functional complexity of domain models.

*Assumption 3.* Based on each DM from a set D, an appropriate generated code framework can be obtained, so there is such a mapping φ:

$$\varphi : D \rightarrow GCF, \tag{5}$$

where GCF (GeneratedCodeFramework) is a set of program code frames that can be used for the construction of SPLs.

*Assumption 4.* All generated code frameworks from a set GCF have different code reusability extents (CRE), so there is such a mapping σ

$$\varphi : GCF \rightarrow CRE, \tag{6}$$

where CRE is a set of possible values of program code reusability extent.

*Assumption 5.* The efficiency coefficient of a certain DMM usage in SPL development can be defined as the ratio of the code reusability extent received using this DM, to the level of its structural complexity, namely:

$$(K)_{Eff}\left[(DM)_{i,j}, (GCF)_{i,j}\right] = \frac{(CRE)_{i,j}}{(DMC)_{i,j}} \tag{7}$$

where $(K)_{Eff}$ is an efficiency coefficient, and the variables $(CRE)_{i,j}$ and $(DMC)_{i,j}$ are defined by expressions (6) and (4).

### 3.3 Analysis of Relationship Between Software Quality Attributes, Complexity Metrics and Rate of Software Code Reuse

We have already shown, that the DDD approach for software development assumes a reuse of different project artefacts, which improves software quality attributes. A variety of projects asset types and their complex and hardly formalized relationships make quantitative analysis practically infeasible. Therefore, it defines a sophisticated and actual task in software engineering (see Sommerville 2011). In order to structure reusable artefacts and perform their qualitative analysis, the article proposes to use mind maps (Guerrero and Ramos 2015), which unlike more formalized approaches (such as UML, IDEF0, etc.) allows to represent

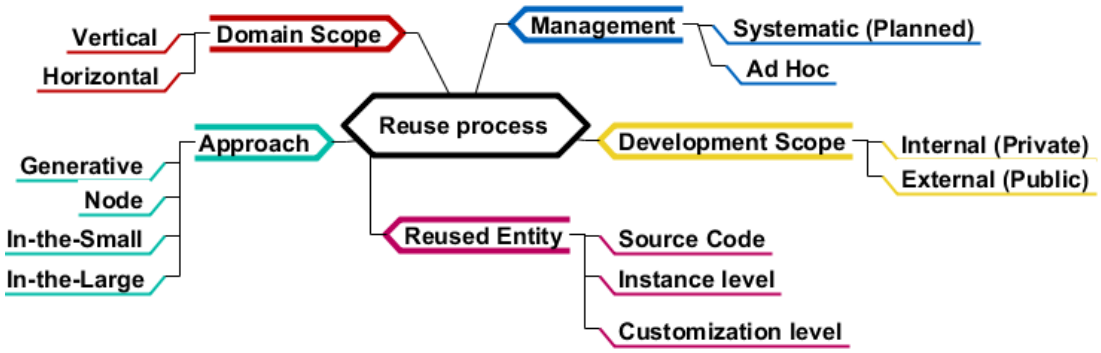



*Figure 4: Software reuse artefacts.*

in any UoD conceptual entities and their semantic relationships of any nature. The ideas about general classification and analysis of software reuse factors may be concluded from research performed in (Paliwal and Shrivastava 2014) and they are depicted as the mind map in Figure 4.

The following entities and their relationships, which qualitatively represent the software reuse process, are shown: *Development Scope* defines the source of reusable components (e.g., from the same project or not); *Approach* defines technical methods to be applied for the software reuse implementation; *Domain Scope* defines where software reuse takes place (e.g., within the same software family or between different ones); *Management* defines how a systematically software reuse process occurs; *Reused entity* defines the object type to be reused.

The next step in this research is to perform a qualitative analysis of relationships between software reusability and software quality attributes as maintainability, adaptability and understandability, as well as software structural complexity. The corresponding mind map for their qualitative analysis is presented in Figure 5. We may conclude that aforementioned software quality attributes, and therefore its ability for reuse, significantly depends on structural software complexity. For an object-oriented approach, it may be defined with the help of well-known metrics (Tkachuk et al. 2016a) as depicted on Figure 5.

Moreover, Preschern et al. (2014) define a correlation between the rate of software reuse and metrics such as DIT, RFC, NOC, CBO and WMC, where:

- DIT (Depth of Inheritance Tree) is defined as the longest path to the current class from the parent class in the class hierarchy.
- RFC (Responses For a Class) is the number of methods, which may be called from an object of the class.
- NOC (Number Of Children) is the number of direct subclasses of the class.
- CBO (Coupling Between Object classes) shows interaction between objects and defines a number of other related classes excluding subclasses.
- WMC (Weighted Methods per Class) is defined as a sum of all class methods, where each method is assessed by its cyclomatic number.

We have to note (Tkachuk et al. 2016a), that there is still a lack of comprehensive analysis in which way the different DMMs influence the complexity of source program code generated basing on an appropriate DM within the DDD software development. Therefore, it is crucial task to identify this correlation in order to reduce costs for DDD-oriented software projects and especially

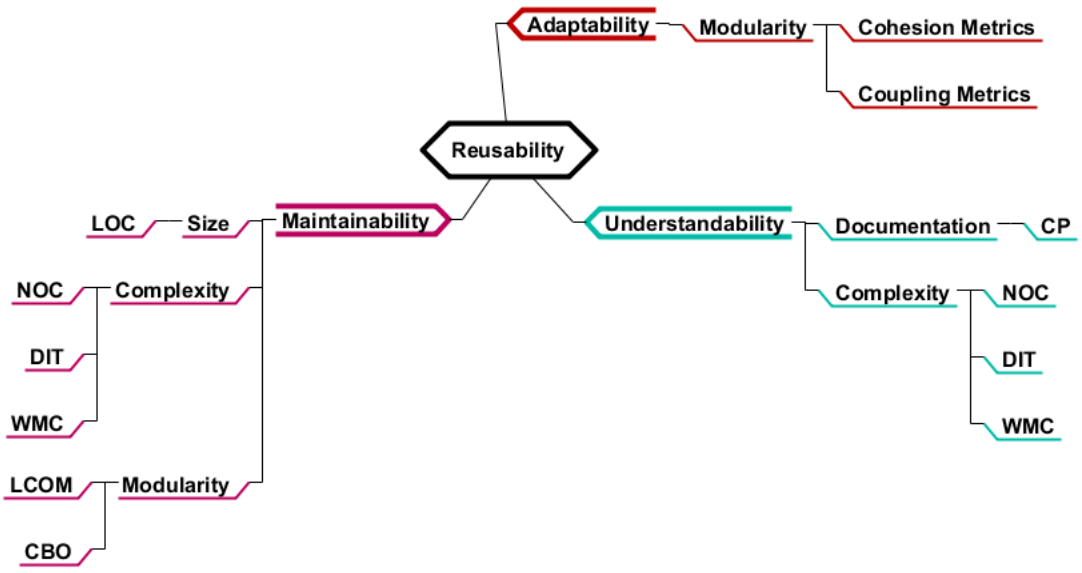



*Figure 5: Quality attributes, metrics and their influence on the software reuse process*

for SPL development. Consequently, taking the analysis results of relationships between quality attributes into account, complexity metrics and software code reuse artefacts shown in Figure 4 and Figure 5, we are able to provide the SPL development approach. It utilizes methods and tools for domain modelling support and uses a source code reusability extent as an effectiveness measurement criteria.

### 3.4 Cybernetic-based Technological Scheme for Software Products Line Development with Usage of DMM

The proposed schema for constructing a SPL, structured as an automated control system with a feedback loop, is finally given on Figure 6. Its main functional modules cooperate in the following manner:

- an initial description of a given UoD in the form of business requirements (as User Stories) to functionality of a target SPL serves as informational basis for building a DM on the conceptual level;
- DMMs (e.g., FODA, JODA, ODM, etc.) and appropriate CASE-tools or domain modelling tools (DMT), such as e.g. FeatureIDE, Actifsource etc., allow to provide a DM realization (DMR);
- a generated code framework (GCF) can be derived based on DMR, and after some changes (e.g., via code refactoring, applying code patterns, etc.) it should be used to build components of a target SPL;
- an efficiency coefficient (see formula (7)), which is used in the feedback control loop, allows us to analyse the domain modelling quality, and to make decisions how to choose the appropriate DMM and CASE-tool for the development of a target SPL.

It is worth to note, that the schema proposed in Figure 6 allows us also to use also other metrics than the proposed efficiency coefficient, in order to perform the analysis of various SPL implementations, and therefore may be seen as improvement of existing methods for solving variability issues in

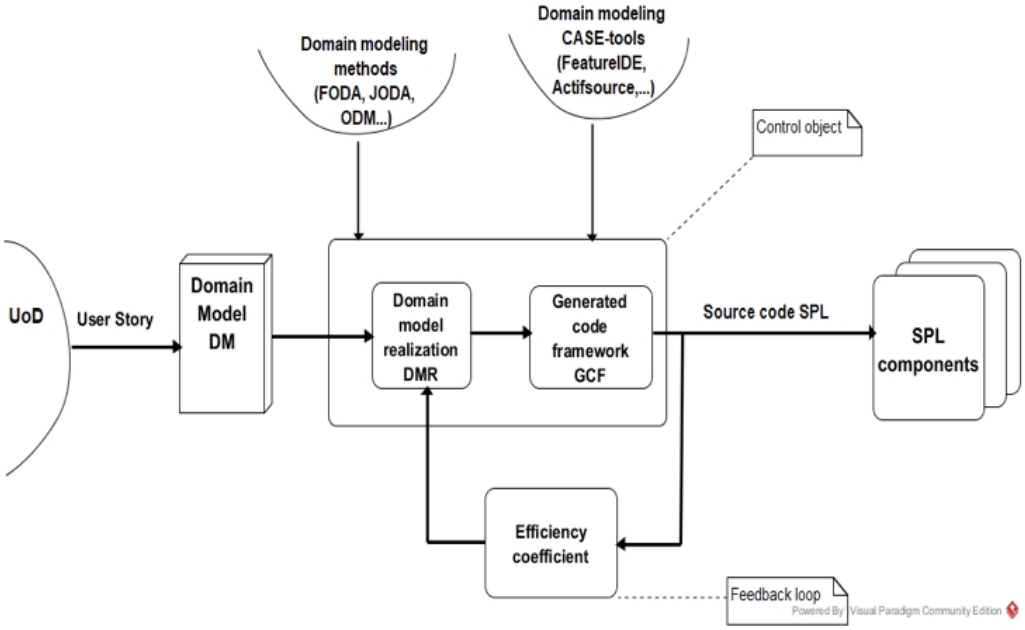



*Figure 6: The proposed technological scheme.*

software development with usage of an alternative DMM.

## 4 Algorithmic Model for Effectiveness Assessment of Domain Modelling Effectiveness

### 4.1 Definition of the Proposed Algorithmic Model

The functional dependence given with formula (7) can not be obtained analytically, therefore, for its investigation, it is necessary to develop a collection of information models, methods and metrics. The corresponding algorithmic model (AM) allows us to present all these components in a formalized way, and it can be presented as follows:

$$AM = \langle InfoBaseWorkFlow(Methods), Metrics \rangle \quad (8)$$

where InfoBase is an information basis of the AM, WorkFlow(Methods) is a set of algorithms (WorkFlow) that implement the appropriate methods (Methods) for assessing the effectiveness of DMM usage, and finally Metrics is a collection of metrics that are used in the corresponding algorithms of the AM model. The information basis of the algorithmic model AM can be presented in the form of the following tuple:

$$InfoBase = \langle M, T, D \rangle \quad (9)$$

where M is a set of methods of DM given in formula (1), T is a set of implementation technologies from formula (2), and D is a set of DM given in formula (3).

The collection *Methods* for evaluating the effectiveness in expression (8) includes:

- the method to determine the code reuse extent of the program code, which is proposed in Tkachuk et al. (2016a);
- the method to estimate the complexity level of a DM, which is given by the expression (4) and is discussed in more detail below.

Finally, the collection *Metrics* from expression (8) consists of:

- metrics for evaluating the structural complexity of the program code (Paliwal and Shrivastava 2014), which are used to analyse the frames generated with the help of the corresponding DM from expression (5);
- metrics for assessing the code reusability extent (Nandakumar 2016), that should be used to construct the corresponding SPL;
- metrics for complexity estimation of the DM (Preschern et al. 2014; Poels et al. 2001) from the set D given by expression(4);
- metrics to define an effectiveness coefficient of the DMM usage in the SPL development or maintenance given by expression (7).

The next step in the proposed approach is the direct development of a method for determining the complexity of DM.

## 4.2 Method for Estimation of Domain Model Complexity

Current publications show, that the problem of DM structural complexity assessment is very relevant both for stand-alone software systems, and for SPL and SSF development. Thus, in Preschern et al. (2014) the approach for an evaluation of DM complexity is proposed, that takes all its main structural components into account, namely:

- a number of DM objects types (#OT),
- a number of links between these objects (#ED), and
- a number of operations defined on them (#DO).

Further, the overall DM structural complexity index CD is considered as the ratio of existing associations to the number of object types: CD = (#ED) / (#OT), but at the same time the number of operations (#DO) is not taken into account while calculating CD and becomes a separate indicator of the DM complexity.

So, this approach does not allow to evaluate comprehensively the corresponding DM complexity. In Poels et al. (2001) a DM complexity problem is considered more detailed. Based on the methodology GOPPRR (Graph-Object-Property-Port-Role-Relationship) for evaluation of individual structural units of a DM, the authors offer their expressions for quantitative estimations:

$$\begin{gathered}(C)_{interface} = \#Relationship + \#Role + \\ + \#Constraints, \\ (C)_{element} = \#Objects + \#Ports, \\ (C)_{property} = \#Properties,\end{gathered} \tag{10}$$

where $(C)_{interface}$ , $(C)_{element}$ , $(C)_{property}$ are the quantitative indicators of complexity for structural DM elements. For a final evaluation of DM complexity the use of the following expressions is suggested:

$$\begin{gathered}(C)_{Overall} = (C)_{interface} + (C)_{element} + \\ + (C)_{property}\end{gathered} \tag{11}$$

However, at the same time this method does not take the functional DM complexity into account, but evaluates only its structure complexity.

Taking aforementioned disadvantages of existing methods for DM complexity assessment into account, it is necessary to propose an integrated approach to structural and functional model complexity estimation, which will allow to obtain a single integral index for this purpose.

In Tkachuk et al. (2016b) an approach for complexity of architectural model assessment in software systems is defined, that was developed using post object-oriented technologies (POOT). It provides a possibility to build analytical expressions to assess the common complexity of different POOT architectures based of their specific components and corresponding weight coefficients, which can be obtained with elaborated expert-oriented procedures. In our opinion, the similar approach can be applied to DM complexity assessment as well. In this way, we propose to introduce

formalized definitions for all major artefacts while constructing an appropriate DM. The small example of such a DM (as a fragment) is shown in Figure 7.

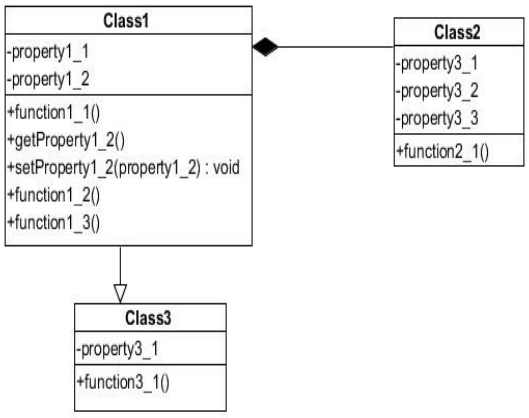



*Figure 7: A fragment of a DM to be assessed*

*Definition 1.* A Domain Model (DM) is a tuple:

$$DM = \langle C, R \rangle \tag{12}$$

where C is a set of classes, and R is a set of relationships between them.

*Definition 2.* Any class $c_k \epsilon C$ is a tuple:

$$c_k = \langle P^{(k)}, F^{(k)} \rangle \tag{13}$$

where $P^{(k)} = \left\{ p_i^{(k)} \right\}, i = \overline{1,n}$ is a set of class''s properties; $F^{(k)} = \left\{ f_j^{(k)} \right\}, j = \overline{1,m}$ is a set of functions (or methods) of a given class, which represent its functional content.

*Definition 3.* A set of links R between elements of the DM is a tuple:

$$R = \left\{ (r)_{i,j} \right\} \tag{14}$$

where $r_{i,j}$ is the connection between classes $i$ and $j$ in this DM. At the same time, each element of the set R refers to one possible element in the set RT.

*Definition 4.* A set RT defines all types of relationship which can be defined between different classes:

$$RT = \{As, Ag, Cm, In\} \tag{15}$$

where As is the name of the Association, Ag is the name for the Aggregation, Cm is the name for the Composition, and In is the name for Inheritance relationships in a given DM.

In (Kang et al. 2004) a quantitative analysis was carried out to estimate the impact of these types of relationships on an objects behaviour in the corresponding DM. Analytically, this dependence can be expressed in the following way:

$$(H)_{As} < (H)_{Ag} < (H)_{Cm} < (H)_{In} \tag{16}$$

where $(H)_{As}, (H)_{Ag}, (H)_{Cm}, (H)_{In}$ are the coefficients of the influence degree for the corresponding type of communication on the overall DM complexity. The proposed method realizes the mapping $\rho$ (see the expression (4)), that in turn supposes to provide the following calculations:

1) the complexity value for each class included in a given DM,
2) the complexity value for all relationships between classes in this DM,
3) the overall complexity value for a given DM.

The appropriate weight coefficients for each type of DM components to be assessment can be determined in an expert-centred way, e. g. using the analytical hierarchy method (Saaty 2000). The steps (1)-(3) are presented below in more details.

*Step 1. Calculation of the complexity for each class.* As the influence of functional class properties on the overall DM complexity is more important, the complexity of each DM class can be obtained using the following expression:

$$CC = 0.3 \times PC + 0.7 \times FPC, \tag{17}$$

where CC (ClassComplexity) is a parameter that determines quantitatively the overall structural and functional class complexity; PC (PropertyComplexity) is a parameter that determines the

complexity of class properties (structural complexity); FPC (FuncPropertyComlexity) is a parameter that determines the complexity of functional properties of a given class (the behavioural complexity of a class). Similarly, a quantitative impact on class complexity (its atomic attributes), and their collections cen be represented by the following expression:

$$PC = 0.4 \times \#STP + 0.6 \times \#CTP, \quad (18)$$

where #STP (SimpleTypeProp) is a number of properties counted for any "simple" types (no collections), #CTP (CollectionTypePrperty) is the number of properties counted for any collection types. The definition of any DM provides its functions signature only (without their implementation details). For this reason, it is enough to calculate their number for the determination of their impact on the overall separate class complexity. By this fact, the following expression is offered:

$$FPC = \#FP, \quad (19)$$

where #FP (FunctionalProperty) is the number of functional properties of the class (operations).

*Step 2. Calculation of the complexity for each relationship* According to expression (14) following weighting factors for each DM class connection types can be determined:

$$(W)_{As} = 0.1, (W)_{Ag} = 0.2, \\ (W)_{Cm} = 0.3, (W)_{In} = 0.4 \quad (20)$$

Given the expression coefficients from (18) the overall connections complexity of a certain DM can be calculated using the following expression:

$$RC = (W)_{As} \times (\#As) + (W)_{Ag} \times (\#Ag) + \\ + (W)_{Cm} \times (\#Cm) + (W)_{In} \times (\#In) \quad (21)$$

where RC (RelationalComplexity) is a parameter, that determines the overall link complexity of DM; #As (AssociationRelation) is the number of association type connections; #Ag (AggregationRelation) is the number of aggregation connections, #Cm (CompoistionRelation) is the number of composition connections; and #IR (InharitanceRelation) is the number of inheritance connections.

*Step 3. Structural and functional DM complexity Calculation.* For the final evaluation of structural and functional DM complexity, we suggest the following expression:

$$DMC = 0.7 \times \#Class \times CC + 0.3 \times RC \quad (22)$$

where DMC (DomainModelComplexity) – is a parameter, that represents the overall structural and functional DM complexity and #Class – an amount of DM classes. The analytical expressions (12) – (22) define an algorithm for implementing a method which determines structural and functional DM complexity, which is a component of the algorithmic model AM given by expression (8).

### 4.3 Method for the Assessment of Code Reusability Extent

In Tkachuk et al. (2016a), an approach for the assessment of code reusability extent (CRE) is proposed, which is generated using DMM and appropriate CASE-tools. This method has three main phases, namely: 1) elaboration of a DM based on the UoD description using initial business requirements, e.g. given in form of user stories; 2) evaluation of source code complexity by using the OOP code metrics mentioned before; 3) calculation of a target CRE in form of their summarized values with weighted coefficients, which have to be defined using, e. g. the analytical hierarchy method. As a result, the following analytical expression was obtained to determine the integrated value of CRE:

$$CRE = 0.12 \times WMC + 0.04 \times RFC \\ + 0.27 \times DIT + 0.36 \times NOC + \\ + 0.21 \times CBO \quad (23)$$

where WMC, RFC, DIT, NOC and CBO are the OOP code complexity metrics mentioned before (see in section 3.3 for more details).

## 5 Feasibility Study of the Proposed Approach: Implementation Workflow and Experimental Results Discussion

### 5.1 Workflow for the Implementation of the Proposed Approach

Based on the algorithmic model represented by expression (8), taking the collection of its components defined by (9) – (14) into account, and using the method for structural and functional DM complexity evaluation, which was presented by (15) – (22), it is possible to elaborate a procedure for the determination of an effectiveness coefficient of the DMM usage given in formula (7). This procedure is presented in form of an UML activity diagram in Figure 8.

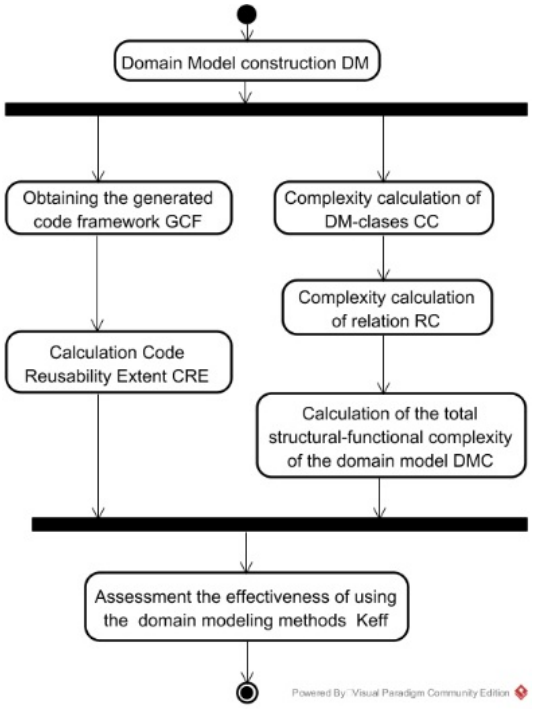



*Figure 8: Workflow for the proposed approach.*

According to expression (7), for the definition of $(K)_{Eff}$ it is necessary to calculate two variables: CRE and DMC respectively. The activities needed to calculate the DMC parameter is described in the previous section. The activity «Obtaining generated code framework GCF» realizes the mapping $\varphi$(see expression (5)), and it can be provided with help of appropriate domain modelling tools (see in Section 2.3). Further, the activity «Calculation of code reusability extent CRE» (see expression (6)) realizes the mapping $\sigma$, using the appropriate metrics given in formula (21).

### 5.2 Use case Domain Models and Experimental Results Discussion

To test the proposed approach, two use case DMs were developed for the UoD «Students Personal Data Processing in Education Management System» using two alternative DMMs: ODM (Organizational Domain Modeling) and JODA (Joint integrated avionics Object oriented Domain Analysis), which can be implemented using CASE - tools as EMF (Eclipse Modeling Framework) and Actifsource (see sections 2.2 and 2.3). The example of generated Java-source code in Figure 9, and the DM elaborated for JODA / Actifsource technology is presented in Figure 10.

This Java - code contains two packages of classes: student.javamodel (provides the implementation and interfaces) and student (realizes the additional resource classes).

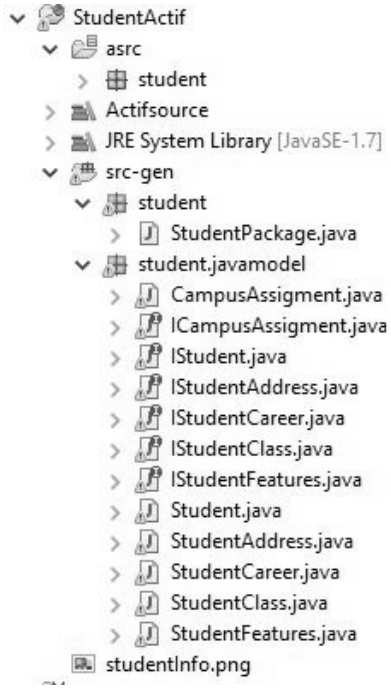



*Figure 9: Generated code framework (GCF).*

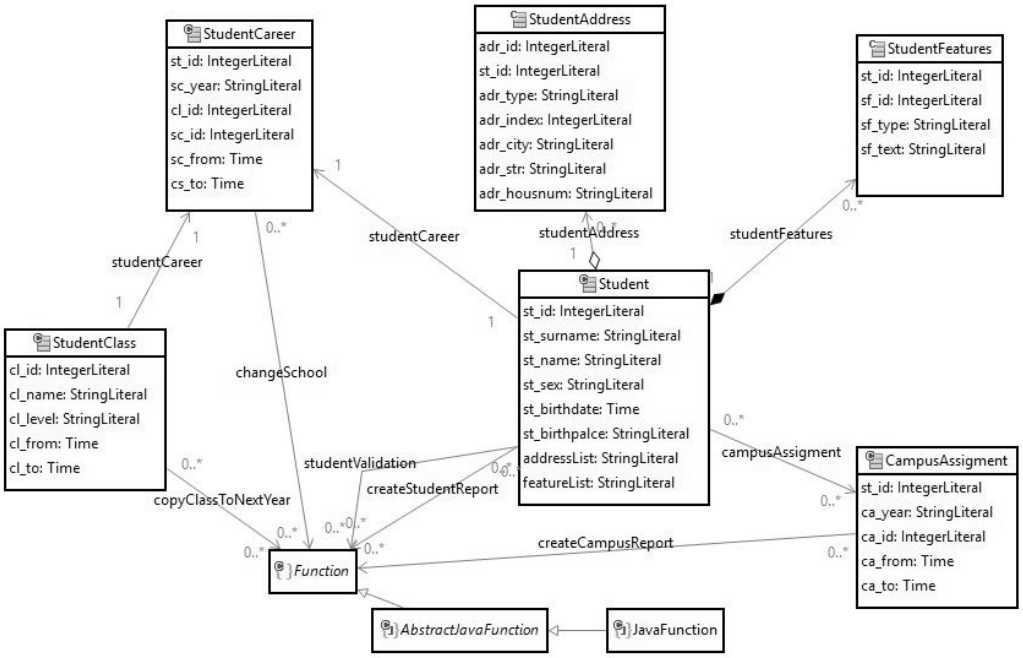



*Figure 10: The example of JODA / Actifsource domain model*

According to the proposed procedure for efficiency coefficient calculation (see Figure 8), it is necessary to get such DM characteristics as DMC and CRE: see formulas (22) and (23). The appropriate experimental data obtained based on the elaborated DM using the JODA / Actifsource tool are presented in Table 2 and Table 3, and for the ODM / EMF tool in Table 4 and Table 5.

*Table 2: DM classes complexity assessment for JODA /Actifsource realization*

| | Class | #STP | #CTP | #FP |
|---|---|---|---|---|
| 1 | Student | 6 | 2 | 2 |
| 2 | StudentAddress | 7 | 0 | 0 |
| 3 | SturdentFeature | 4 | 0 | 0 |
| 4 | StudentCareer | 6 | 0 | 1 |
| 5 | StudentClass | 5 | 0 | 1 |
| 6 | CampusAsignement | 5 | 0 | 1 |
| 7 | Function | 0 | 0 | 0 |
| 8 | AbstractFunction | 0 | 0 | 0 |
| 9 | JavaFunction | 0 | 0 | 0 |
| | $\sum$ | **33** | **2** | **5** |

*Table 3: DM relationships complexity assessment for JODA / Actifsource implementation*

| | |
|---|---|
| Association | 8 |
| Aggregation | 1 |
| Composition | 1 |
| Inheritance | 2 |

*Table 4: DM classes complexity assessment for ODM / EMF implementation*

| | Class | #STP | #CTP | #FP |
|---|---|---|---|---|
| 1 | Student | 6 | 2 | 2 |
| 2 | StudentAddress | 7 | 0 | 0 |
| 3 | SturdentFeature | 4 | 0 | 0 |
| 4 | StudentCareer | 6 | 0 | 1 |
| 5 | StudentClass | 5 | 0 | 1 |
| 6 | CampusAsignement | 5 | 0 | 1 |
| | $\sum$ | 33 | 2 | 5 |

*Table 5: DM relationships complexity assessment for ODM / EMF implementation*

| | |
|---|---|
| Association | 3 |
| Aggregation | 0 |
| Composition | 2 |
| Inheritance | 0 |

Based on the parameters presented in the Tables 2-5 and using the formulas (16) – (22) the following values for the elaborated DM were obtained:

**for JODA / Actifsource realization**

$PC(JODA/Actifsource) = 0.4 * 33 + 0.6 * 2 = 14.4$
$FPC(JODA/Actifsource) = 5.0$
$CC(JODA/Actifsource) = 0.3 * 14.4 + 0.7 * 5 = 7.82$
$RC(JODA/Actifsource) = 0,1 * 8 + 0.2 * 1 + 0,3 * 1 + 0,4 * 2 = 2,1$
$DMC(JODA/Actifsource) = 0.7 * 9 * 7.82 + 0.3 * 2.1 = 49,9$

**for ODM/EMF realization**

$PC(ODM/EMF) = 0.4 * 33 + 0.6 * 2 = 13.2 + 1.2 = 14.4$
$FPC(ODM/EMF) = 5.0$
$CC(ODM/EMF) = 0.3 * 14.4 + 0.7 * 5 = 4.32 + 3.5 = 7.82$
$RC(ODM/EMF) = 0,1 * 3 + 0.2 * 0 + 0,3 * 2 + 0,4 * 0 = 0,9$
$DMC(ODM/EMF) = 0.7 * 6 * 7.82 + 0.3 * 0.9 = 33.11$

At the same time, it is necessary to get the values of code reusability extent (CRE) for these two alternative DMs. To calculate CRE, formula (23) should be used after the appropriate OOP metrics were defined (see Tkachuk et al. 2016a for more details), and the following values of the CRE were obtained:

$$CRE(JODA/Actifsource) = 7.78$$
$$CRE(ODM/EMF) = 16.67$$

Using expression (7), the final values of efficiency coefficients for the tested DMs are calculated (see Table 6).

In graphical representation, these results are shown in Figure 11.

*Table 6: Efficiency coefficients values obtained*

| | CRE | DMC | $(K)_{eff}$ |
|---|---|---|---|
| JODA / Actifsource | 7,78 | 49.90 | 0.156 |
| ODM / EMF | 16,67 | 33.11 | 0.503 |

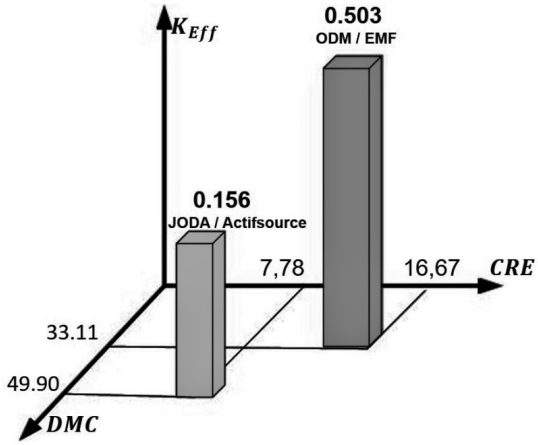



*Figure 11: Graphical interpretation of the results obtained.*

Based on this feasibility study we can conclude, that domain modelling with usage of ODM / EMF technology provides a higher rate of efficiency in comparison with the JODA / Actifsource tool, and these results should be taken into account for domain-driven development of a target software product line.

## 6 Conclusions and Future Work

In this paper we have considered essential aspects of domain-driven development (DDD), and especially, the usage of domain modelling methods (DMM) in modern software engineering with respect to the improvement of software quality, and as a way to develop software product lines (SPL). To emphasize advantages of DDD and DMM usage, a 3-level design scheme is proposed, that reflects variability issues in this framework. According to this metaphor, the main attention is paid to the phases of logical domain modelling level and to physical modelling, with usage of two alternative DMM-methods: JODA and ODM approaches. The algorithmic model for an efficiency coefficient estimation of alternative DMM

usage is proposed, which utilizes structured data resources, expert methods and metrics used in the SPL development process. A feasibility study for the proposed approach was performed, and the obtained experimental results are shown, which can be useful in the development of a target SPL.

In future, we are going to construct a more sophisticated collection of software complexity metrics, e. g. to take the metrics of relationships between packages into account, and to improve the prototype of our software tool to support the proposed evaluation approach.

## Acknowledgments

The authors, former visiting researchers and students at Alpen-Adria Universität Klagenfurt in Austria, would like to express their special gratitude to Prof. Dr. Dr. h.c. Heinrich C. Mayr for stimulating discussions of problems related to the topics presented in this paper.

We would like also to thank students of National Technical University "Kharkiv Polytechnic Institute" A. Tkachuk and M. Raldugyn for their support in software implementation to test our approach.

## References

Batini C., Ceri S., Navathe S. (1992) Conceptual Database Design: An Entity-Relationship Approach. Benjamin Publishing Company

Bayer J., Kettemann S., Muthig D. (2004) Principles of software product lines and process variants. Process Family Engineering in Service-Oriented Applications.BMBF-Project. In: PESOA-Report No. 03/2004.

Capilla R., Bosch J., Kang K. (2013) Systems and Software Variability Management. Springer

Evans E. (2004) Domain-Driven Design Tackling Complexity in the Heart of Software, 1st ed. Prentice Hall

Ferré X. (1999) An Evaluation of Domain Analysis Methods. In: 4th CAiSE.IFIP8.1 International Workshop in Evaluation of Modeling Methods in Systems Analysis and Design, pp. 1–13

Greenfield J., Short K. (2004) Software Factories: Assembling Application with Patterns, Models, Frameworks and Tools. Wiley, Indianapolis

Guerrero J., Ramos P. (2015) Mind Mapping for Reading and Understanding Scientific Literature. In: International Journal of Current Advanced Research. 4(11), pp. 485–487

Kang D., Xu B., Lu J. (2004) A Complexity Measure for Ontology based on UML // Distributed Computing Systems. In: Proceedings of the 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, pp. 222–228

Karagiannis D., Mayr H., Mylopoulos J. (2016) Domain-Specific Conceptual Modeling: Concepts, Methods and Tools. Springer

Kelly S., Tolvanen J. (2008) Domain-Specific Modeling: Enabling Full Code Generation. Wiley Computer Society Press

Kittlaus H.-B., Clough P. (2009) Software Products: Terms and Characteristics. In: Software Product Management and Pricing.. Springer

Klaus P. (2016) Learning and Evolution in Dynamic Software Product Lines. In: 11th Int'l Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS) Carlo Ghezzi, Sam Malek (eds.)

Lee S. U.-J. (2015) An Effective Methodology with Automated Product Configuration for Software Product Line Development, Article ID 435316. In: Mathematical Problems in Engineering 2015

Michael J., Mayr H. C. (2015) Creating a Domain Specific Modelling Method for Ambient Assistance. In: International Conference on Advances in ICT for Emerging Regions (ICTer2015). IEEE, pp. 119–124

Microsoft Corp Accessed on: 15.05.2017 https://www.microsoft.com/uk-ua/download/office.aspx

Nagesh P., Vivek S. (2016) An Approach to Find Reusability of Software Using Objet Oriented Metrics. In: International Journal of Innovative Research in Science, Engineering and Technology 3(3) Tiwari K. (ed.)

Nandakumar A. (2016) Constructing Relationship between Software Metrics and Code Reusability in Object Oriented Design. In: International Journal of Advanced Computer Science and Applications 7(2)

OMG (2010) OMG Unified Modeling Language, Superstructure. Version 2.3. OMG

Oracle Corp Accessed on: 15.05.2017 http://www.oracle.com/technetwork/middleware/soasuite/overview/index.html

Paliwal N., Shrivastava V. (2014) An Approach to Find Reusability of Software Using Objet Oriented Metrics. In: International Journal of Innovative Research in Science, Engineering and Technology 3 K. T. (ed.)

Patdar S. M. (2014) Literature Survey on Algorithmic Methods for Software Development Cost Estimation. In: Int. J. Computer Technology & Applications 5(1), pp. 183–188

Perovich D., Rossel P., Bastarrica M. (2009) Feature model to product architectures: Applying MDE to Software Product Lines. In: Joint Working IEEE/IFIP Conference on Software Architecture & European Conference on Software Architecture., pp. 201–210

Poels G., Dedene G., Viane S. (2001) A Quantitative Assessment of Complexity Of Static Conceptual Schemata For Reference Types Of Front-office. In: In Proceedings of the Fifth International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE2001)

Pohl K., Bockle G., Van Der Linden F. (2010) Software product line engineering: Foundations, Principles, and Techniques. Springer

Preschern C., Kajtazovic N., Kreiner C. (2014) Evaluation of Domain Modeling Decisions for two identical Domain Specific Languages. In: Lecture Notes on Software Engineering 2(1), pp. 37–41

Reinhartz-Berger I. (2013) Domain Engineering: Product Lines, Languages, and Conceptual Models. Springer

Saaty T. (2000) Fundamentals of the Analytic Hierarchy Process. RWS Publishing

Software Product Line Conference Accessed on: 15.05.2016 http://splc.net/

Sommerville I. (2011) Software Engineering. Addison-Wesley, Boston, MA

Tkachuk M., Martinkus I., Gamzayev R., Tkachuk A. (2016a) An Integrated Approach to Evaluation of Domain Modeling Methods and Tools for Improvement of Code Reusability in Software Development. In: Mayr H. C., Pinzger M. (eds.) INFORMATIK 2016. Lecture Notes in Informatics (LNI) Vol. P-259. Kollen Druck, Verlag GmbH, pp. 143–156

Tkachuk M., Nagorniy K., Gamzayev R. (2016b) Models, Methods and Tools for Effectiveness Estimation of Post Object-Oriented Technologies in Software Maintenance. In: ICTERI 2015: Revised Selected Papers, Series title: Communications in Computer and Information Science. 594 Yakovyna V. (ed.), pp. 20–37

Tune N., Millet S. (2015) Patterns, Principles And Practices Of Domain-driven Design, 1st ed. John Wiley & Sons

# Consistency Management Techniques for Variability Modelling

Alexander Felfernig*,[a], Thomas Gruber[a], Martin Stettinger[a]

[a] Graz University of Technology, Institute for Software Technology, Applied Software Engineering, Graz, Austria.

Abstract. *Feature models are a means to represent software variability. Due to their logical grounding, such representations allow for automated reasoning about specific model properties. In this article, we show how the concepts of conflict detection and model-based diagnosis can be applied to analyse and improve the quality of a feature model. The example feature model used in this context is based on the variability information of a real-world event management environment.*

Keywords. Variability Modelling • Feature Models • Consistency Management • Configuration

## 1 Introduction

Requirements Engineering (RE) is considered as a crucial phase in software development (Kop and Mayr 1998; Leffingwell and Widrig 2003; Mayr and Kop 2002; Mayr et al. 2007). Low quality requirement models can trigger enormous follow-up costs manifested, for example, in terms of re-design needs, re-implementation, debugging, and testing. A specific task in the RE context is to model software variability (Benavides et al. 2010; Kang et al. 1990; Thum et al. 2009), i. e., which combinations of components are allowed when delivering a software. Feature models are an approach to variability modelling especially developed to support the construction of software product lines (Kang et al. 1990). Due to the increasing complexity of feature models, the identification of inconsistencies becomes a challenging task. In this article, we discuss concepts that support the engineering of feature models on the basis of consistency-based analysis approaches, more specifically, conflict detection (Junker 2004) and model-based diagnosis (Reiter 1987). These approaches support the automated identification of inconsistencies in feature models and can be used for further purposes such as *automated testing and debugging*, *redundancy detection*, and software quality improvement related activities (Felfernig et al. 2014).

Feature models can be distinguished with regard to the degree of expressiveness of feature representations and corresponding constraints (Benavides et al. 2010). *Basic feature models* (Kang et al. 1990) allow the definition of basic relationships between features, for example, the inclusion of a specific feature *x excludes* another feature *y*. *Cardinality-based feature representations* (Czarnecki et al. 2005) additionally allow the definition of cardinalities of relationships (in basic models, the upper bound of cardinalities is 1). Finally, *extended feature models* allow the definition of feature properties represented as feature attributes (Batory 2005). Without loss of generality, our examples are based on *basic feature models*. The presented concepts can also be applied to the advanced feature model types introduced in (Batory 2005; Czarnecki et al. 2005).

Designing feature models is an error-prone activity that results from a cognitive overload of engineers engaged in the development and maintenance of such models (Benavides et al. 2013). Artificial Intelligence (AI) techniques can support the identification of different types of inconsistencies in an automated fashion (Benavides et al. 2010). Inconsistency identification and resolution

* Corresponding author.
E-mail. alexander.felfernig@ist.tugraz.at

techniques for feature models have been introduced, for example, by (White et al. 2010) where dead features are interpreted as a model anomaly (a feature part of a model that can never be included in a configuration). In this context, feature models are translated into corresponding constraint-based representations (Tsang 1993) and the identification of dead features is directly encoded as a constraint satisfaction problem. An overview of different types of model anomalies that can occur in feature models is provided in (Benavides et al. 2010). These anomalies are also the basis for the discussions in this article.

Feature models can be regarded as specific type of configuration models (Felfernig et al. 2014) represented, for example, in terms of constraint satisfaction problems (Tsang 1993). The diagnosis of inconsistent constraint-based representations has first been introduced in (Bakker et al. 1993) where inconsistencies in constraint models are resolved on the basis of the concepts of model-based diagnosis (Reiter 1987). This work has been extended to test scenarios where diagnoses are calculated on the basis of a given set of test cases (Felfernig et al. 2004). This approach is based on the idea of the identification of minimal conflict sets (Junker 2004) and their resolution based on the concept of a Hitting Set Directed Acyclic Graph (HSDAG) (Reiter 1987). More recent approaches to model-based diagnosis omit the calculation of conflict sets and directly determine minimal diagnoses (see, e.g., Felfernig et al. 2018).

In this article, we focus on aspects related to the identification of erroneous constraints in feature models. In this context, our major contributions are the following. On the basis of a formalization of feature models as constraint satisfaction problems, we discuss different types of inconsistencies that can occur in feature model development. Second, we show how these inconsistencies can be localized on the basis of the concepts of conflict detection and model-based diagnosis. Finally, we discuss open issues for future research. Our examples are based on a feature model derived from a commercial event management software.

The remainder of this article is organized as follows. In Section 2, we introduce a feature model from the EVENTHELPR[1] environment. This model serves as a working example throughout this article. In Section 3, we sketch logic approaches to detect and resolve inconsistencies. In Section 4, we introduce and formalize different types of inconsistencies that can occur when developing feature models. Ways to resolve inconsistencies in our example feature model are discussed in Section 5. Section 6 provides a discussion of different open research issues. This article is concluded with Section 7.

## 2 Example Feature Model and Semantics

A feature model defines a complete set of allowed configurations, i.e., the combinations of features that can be jointly included in a configuration. The modelling concepts that can be used to build feature models are: *features*, *relationships between features* (represented as constraints), and so-called *cross-hierarchy constraints* (Kang et al. 1990). For a detailed discussion of feature modelling techniques we refer to (Batory 2005; Czarnecki et al. 2005; Kang et al. 1990).

Since feature models are a basic mechanism to define variability properties, we can interpret these models as configuration knowledge bases (Felfernig et al. 2014). Features are the structural elements of feature models. They can be either *included in* or *excluded from* a specific configuration. Each feature is associated with the domain {*true*, *false*} where *true* specifies *feature inclusion* (into a corresponding configuration) and *false* defines *feature exclusion*. Features are connected via constraints (partially defined by relationships) that specify additional restrictions on possible combinations of features. In the following, we introduce six different types of constraints that are typically used in feature modeling: *mandatory*, *optional*, *alternative*, *or*, *requires*, and *excludes*.

The formalization of these constraints is a basis for performing automated reasoning on feature model properties. Feature configuration tasks can

[1] www.eventhelpr.com.

be formalized as *Constraint Satisfaction Problems* (*CSPs*) (Tsang 1993) as follows (see Definition 1).

*Definition 1 (Feature Configuration Task).* A *feature configuration task* can be defined by a triple $(F, D, C)$ where $F$ represents a set of features and each feature $f_i \in F$ has an associated domain $dom(f_i) \in D = \{true, false\}$. Finally, $FM \cup CREQ$ represents a set of constraints $c_i \in C$ where $FM$ are domain model constraints contained in the feature model and $CREQ$ is a set of customer requirements used to specify customer-specific requirements with regard to a feature configuration ($FM \cup CREQ = C$).

Based on a feature configuration task definition, different configurations can be determined. A *feature configuration* (solution to a feature configuration task) can be defined as follows.

*Definition 2 (Feature Configuration).* A *feature configuration* is a complete set of value assignments ($val(f_i) \in \{true, false\}$) to features $f_i \in F$. A feature configuration is *consistent*, if the value assignments are consistent with the constraints in $C$. Furthermore, it is regarded as *valid* if it is consistent and *complete* (each variable has a corresponding value assignment).

*Constraint Types.* The following six constraint types are often used to build feature models (Benavides et al. 2010). An example of a feature model using these constraint types is depicted in Figure 1. In the following, we define the semantics of these constraint types.

*Mandatory.* A feature $f_2$ is in a mandatory relationship with feature $f_1$ if whenever $f_1$ is part of the configuration, $f_2$ must be part of the configuration (and vice-versa). On the logical level, this property can be defined in terms of an equivalence, i. e., $f_1 \leftrightarrow f_2$. An example of a mandatory relationship in Figure 1 is the *participation* feature: in any case, the type of participation in an event has to be defined.

*Optional.* A feature $f_2$ is in an optional relationship with feature $f_1$ if whenever $f_1$ is part of the configuration, $f_2$ can be part of the configuration. Vice-versa, if $f_2$ is part of the configuration, $f_1$ must be included. On the logical level, this property can be defined in terms of an implication, i.e., $f_2 \rightarrow f_1$. An example of an optional relationship in Figure 1 is the *content* feature: sending emails and uploading photos is considered as optional in every EventHelpr instance.

*Alternative.* This type of relationship specifies an "xor" semantics, i. e., exactly one of a given set of features has to be included in a configuration. Given a feature $f_1$ and a set of sub-features $\{f_{11}, f_{12}, .., f_{1n}\}$, the alternative relationship can be formalized as follows: $f_1 = true \leftrightarrow ((f_{11} = true \wedge f_{12} = false \wedge ..f_{1n} = false) \vee (f_{12} = true \wedge f_{11} = false \wedge ..f_{1n} = false) \vee ...))$. An example of an *alternative* relationship depicted in Figure 1 are the two *participation subtypes* (*invitation & login*, *no login needed*).

*Or.* This type of relationship specifies an "or" semantics, i. e., at least one of a given set of alternative features has to be included in a configuration. Given a feature $f_1$ and a set of sub-features $\{f_{11}, f_{12}, .., f_{1n}\}$, the *or* relationship can be formalized as follows: $f_1 = true \leftrightarrow f_{11} = true \vee f_{12} = true \vee ..f_{1n} = true$. An example of an *or* relationship depicted in Figure 1 is the inclusion of *notification* mechanisms, i.e., the sub-features of the *notification* feature.

*Requires.* This type of relationship specifies a "requires" semantics, i. e., the inclusion of a feature $f_1$ requires the inclusion of a feature $f_2$ ($f_1 \rightarrow f_2$). An example of a *requires* relationship depicted in Figure 1 is the needed inclusion of *emails* and *postings* as a precondition for the feature *new postings* notification. Requires relationships are regarded as one type of *cross-tree constraint*.

*Excludes.* Such a relationship specifies an "incompatibility" semantics, i. e., the inclusion of a feature $f_1$ excludes a feature $f_2$ (and vice-versa). On a logical level, this property can be specified as $\neg(f_1 \wedge f_2)$. An example of an *excludes* relationship depicted in Figure 1 is the incompatibility of the *interactive agenda* service and scenarios where *no login* is needed. Similar to *requires* relationships, *excludes* constraints are regarded as a type of *cross-tree constraint*.

*Customer requirements* ($CREQ$) are user preferences that should be taken into account by a constraint solver when searching for a solution. In

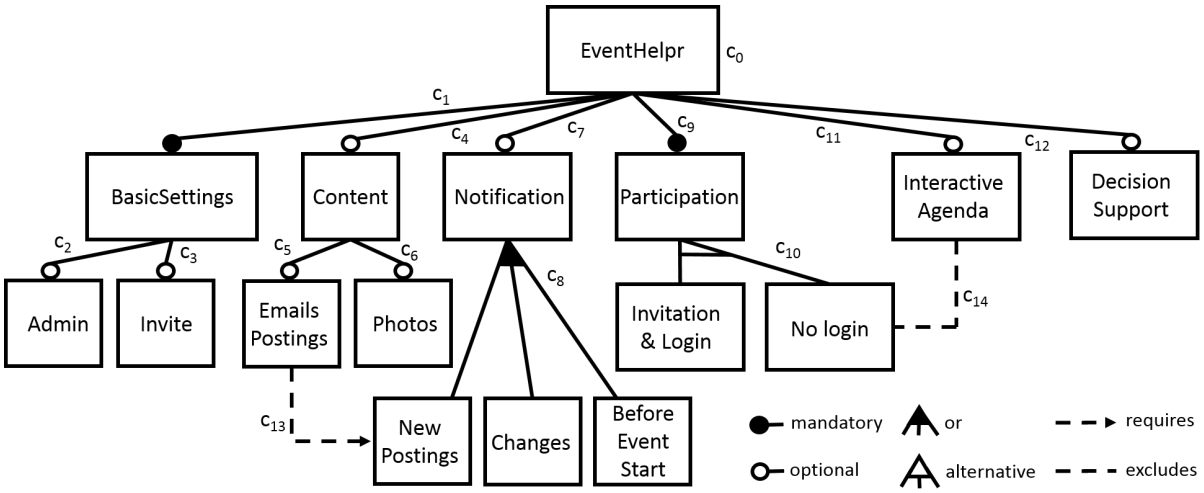



*Figure 1: Variability model of a real-world event management environment (www.eventhelpr.com). The different variants describe ways in which the application can be presented to end-users (dashed arrows specify* requires *and dashed lines* excludes *constraints).*

other words, $CREQ$ specifies those features that should be included in a feature configuration from the user point of view.

An example of parts of a feature model of the EventHelpr environment is depicted in Figure 1. The constraint-based representation (Tsang 1993) that can be derived thereof is the following.[2]

- $F = \{$*eventhelpr (eh), basicsettings (bas), admin (adm), invite (inv), content (con), emailspostings (email), photos (phot), notification (not), newpostings (newpost), changes (ch), beforeeventstart (bef), participation (part), invitationlogin (invit), nologin (nologg), interactiveagenda (intag), decisionsupport (dec)*$\}$.
- $D = \biguplus_{f_i \in F} dom(f_i)$.
- $FM = \{c_0 : eh = true, c_1 : eh \leftrightarrow bas, c_2 : adm \rightarrow bas, c_3 : inv \rightarrow bas, c_4 : con \rightarrow eh, c_5 : email \rightarrow con, c_6 : phot \rightarrow con, c_7 : not \rightarrow eh, c_8 : not \leftrightarrow newpost \vee ch \vee bef, c_9 : part \leftrightarrow eh, c_{10} : part \leftrightarrow (invit \wedge \neg nolog \vee \neg invit \wedge nolog), c_{11} : intag \rightarrow eh, c_{12} : dec \rightarrow eh, c_{13} : email \rightarrow newpost, c_{14} : \neg(intag \wedge nolog)\}$.

A solution (configuration) for the feature model[3] depicted in Figure 1 is: $\{eh = true, bas = true, adm = true, inv = false, con = true, email = false, phot = true, not = true, newpost = false, ch = true, bef = false, part = true, invit = false, nolog = true, intag = false, dec = false\}$.

## 3 Resolving Inconsistencies

In this section, we discuss undesirable properties of feature models which trigger an unintended behaviour in one way or another. We explain such properties in terms of *inconsistencies* on the logical level. Inconsistent models include conflicts (Junker 2004) that are induced *internally* by model constraints (in $FM$) or *externally*, for example, by test cases. Inconsistencies can be explained on the basis of *minimal conflict sets* (Junker 2004) (see Definition 3).

*Definition 3 (Conflict Set).* A *conflict set* $CS \subseteq FM$ is a set of constraints s.t. inconsistent($CS$). $CS$ is *minimal* if $\neg\exists CS'$: conflict set ($CS'$).

*Minimal conflict sets* help to easily resolve conflicts since the deletion of at least one element

[2] The names in brackets are introduced to increase the readability of constraints in $FM$.

[3] Variable assignments are abbreviated, for example, $eh = true$ represents $val(eh) = true$.

from the set guarantees that the conflict is already resolved. A set, that contains all elements needed to delete at least one element from each existing conflict, is denoted as *hitting set* (Reiter 1987). Conflict resolution can be performed on the basis of the concepts of *model-based diagnosis* (Reiter 1987). Following the standard diagnosis approach introduced by (Reiter 1987), *Hitting Set Directed Acyclic Graphs* (*HSDAGs*) have to be constructed where individual paths of the tree represent hitting sets (diagnoses). A diagnosis task can be defined as follows (see Definition 4). For an overview of different conflict detection and related diagnosis algorithms we refer, for example, to (Felfernig et al. 2014).

*Definition 4 (Diagnosis).* A *diagnosis* $\Delta \subseteq FM$ is a set of constraints s.t. consistent$(FM - \Delta)$. $\Delta$ is *minimal* if $\neg\exists\Delta' \subset \Delta$: diagnosis $(\Delta')$.

Typically, we are interested in analysing specific parts of $C$, i. e., either $FM$ if we are interested in faulty constraint parts of the feature model, or $CREQ$ if we are interested in (minimal sets of) customer requirements that induce an inconsistency. Conflict sets and diagnoses are then determined from the individual parts of $C$, i. e., $CS$ is then either a subset of $FM$ or $CREQ$, the same holds for corresponding diagnoses $\Delta$.

In the following, we focus on the first aspect, i. e., constraints in the feature model responsible for an inconsistency (as mentioned in Definitions 3 and 4, conflict sets and diagnoses are composed of constraints in $FM$).[4]

## 4 Inconsistencies in Feature Models

In the following, we discuss feature model properties that make a feature model ill-formed. For each property, we characterize the underlying inconsistency on a formal level and show ways to resolve the inconsistency (see also Table 1). Thus, we try to explain ill-formed model properties in terms of inconsistencies between intended and unintended feature model properties.

*Void feature models.* A *void* feature model is inconsistent per-se, i. e., no solution can be identified for a given feature configuration task $(F, D, FM)$. A diagnosis $\Delta$ for a void feature model $FM$ is defined as $\Delta \subseteq FM : consistent(FM - \Delta)$.

*Test-induced void feature models.* Feature models can be tested on the basis of a test suite $T = \{t_1, t_2, .., t_m\}$ consisting of positive test cases[5] specifying the intended behaviour of a knowledge base. A feature model is consistent with $T$ if $\forall t_j \in T : FM \cup \{t_j\}$ is consistent. If some of the test cases induce conflicts $(CS_k)$, these have to be resolved on the basis of the concepts of model-based diagnosis (Reiter 1987). A diagnosis $\Delta$ for a test-induced void feature model is defined as $\Delta \subseteq FM$ such that $\forall t_j \in T : consistent(\{t_j\} \cup FM - \Delta)$.

*Dead features.* A feature $f \in F$ is regarded as a dead one if it is not part of any of the theoretically possible solutions. More formally, $\{f = true\} \cup FM$ is inconsistent. A diagnosis $\Delta$ for a dead feature is defined as consistent$(FM - \Delta \cup \{f = true\})$.

*Fully mandatory features.* A feature $f$ is fully mandatory, if it is included in every possible configuration, i.e., inconsistent$(\{f = false\} \cup FM)$. If we want to allow configurations with $f$ not included $(f = false)$, a diagnosis $\Delta$ for a fully mandatory feature is defined as consistent$(FM - \Delta \cup \{f = false\})$.

*False optional features.* A feature $f_2$ is *false optional*, if it is modelled as optional with regard to a parent feature $f_1$ but in fact is contained in every configuration $f_1$ is included. A diagnosis $\Delta$ for a false optional feature is defined as consistent$(FM - \Delta \cup \{f_2 = false\} \cup \{f_1 = true\})$, i.e., there exists at least one configuration with $f_2 = false$ and $f_1 = true$.

*Redundant constraints in FM.* A constraint $c_i \in FM$ is regarded as redundant if $FM - \{c_i\} \models c_i$, i.e., $c_i$ logically follows from $FM - \{c_i\}$. Consequently, deleting $c_i$ from $FM$ preserves the semantics of the feature model. In other

[4] Details on diagnosing $CREQ$ can be found in (Felfernig et al. 2004).

[5] For a discussion of the handling of negative test cases we refer to (Felfernig et al. 2004).

words, the solution space of the feature model remains exactly the same. If $FM = \{c_1, .., c_n\}$ is a set of non-redundant constraints of a feature model, and $\widehat{FM}$ is the negation of $FM$ (i. e., $\widehat{FM} = \{\neg c_1 \vee \neg c_2 \vee ... \vee \neg c_n\}$, then $FM \cup \widehat{FM}$ is inconsistent. As a consequence, a redundant constraint set $FM_r$ can be made non-redundant by determining a minimal set of constraints $FM \subseteq FM_r : inconsistent(FM \cup \widehat{FM})$. This way, conflict detection algorithms can be applied to make knowledge bases non-redundant.

*Implicit feature groups*. If two features have exactly the same value in each possible configuration, there is a strong dependency between these features. In some cases, these features can even be combined to reduce the number of needed constraints and thus increase model understandability and maintainability. Two features $f_1$ and $f_2$ form an implicit group if inconsistent($\{f_1 \wedge \neg f_2 \vee \neg f_1 \wedge f_2\} \cup FM$), i. e., there does not exist a solution in which the two features have different values.

## 5 Inconsistencies in Example Model

In order to exemplify the discussed model properties, we introduce an adapted version of the feature model shown in Section 2 (see Figure 2). The constraint-based representation (Tsang 1993) that can be derived thereof, is the following.

- $F = \{$*eventhelpr (eh), basicsettings (bas), admin (adm), invite (inv), content (con), emailspostings (email), photos (phot), notification (not), newpostings (newpost), changes (ch), beforeeventstart (bef), participation (part), invitationlogin (invit), nologin (nolog), interactiveagenda (intag), decisionsupport (dec)*$\}$.
- $D = \mathbb{U}_{f_i \in F} dom(f_i)$.
- $FM = \{c_0 : eh = true, c_1 : eh \leftrightarrow bas, c_2 : adm \rightarrow bas, c_3 : inv \rightarrow bas, c_4 : con \leftrightarrow eh, c_5 : email \leftrightarrow con, c_6 : phot \rightarrow con, c_7 : not \rightarrow eh, c_8 : not \leftrightarrow newpost \vee ch \vee bef, c_9 : part \leftrightarrow eh, c_{10} : part \leftrightarrow (invit \wedge \neg noglog \vee \neg invit \wedge nolog), c_{11} : intag \leftrightarrow eh, c_{12} : dec \leftrightarrow eh, c_{13} : email \rightarrow newpost, c_{14} : \neg(intag \wedge nolog), c_{15} : dec \rightarrow nolog\}$.

*Void feature model*. No solution exists for the feature model defined in Figure 2 since both features, *interactiveagenda (intag)* and *decisionsupport (dec)* are mandatory, *dec* requires *nolog* and *intag* excludes *nolog*. The (singleton) resulting minimal conflict set is $CS_1 : \{c_0, c_{11}, c_{12}, c_{14}, c_{15}\}$. In this example, only one minimal conflict set exists and the deletion of any of these constraints restores consistency, i.e., each individual constraint represents a diagnosis $\Delta$ (e.g., $\Delta = \{c_{14}\}$). We want to emphasize that constraint $c_0$ has a specific role: if it is set to *true*, it is guaranteed that only non-empty feature configurations, i.e., feature configurations with at lest one activated feature, are taken into account. For this reason, $c_0$ has a special role and is in most of the cases not considered a diagnosis candidate.

*Test-induced void feature model*. In the following, we assume that constraint $c_{14}$ in the feature model of Figure 2 has been deleted to restore consistency. An example of a test-induced void feature model is the model depicted in Figure 1 combined with the test-case $t_1 : intag \wedge nolog$, i.e., inconsistent($\{t_1\} \cup FM$). The minimal conflict set is $CS : \{c_{14}\}$ since inconsistent($\{t_1\} \cup \{c_{14}\}$).

*Dead features*. In our example feature model of Figure 2, feature *invitationlogin (invit)* can be considered as dead since it is not possible to include this feature in a configuration. The reason is that feature *decisionsupport (dec)* requires the inclusion of feature *nolog*. Due to the *alternative* relationship between *participation (part)*, *inv*, and *nolog*, it is not possible to include feature *inv*.

*False optional feature*. Since the inclusion of *emailspostings (email)* requires the inclusion of *newpostings (newpost)*, *notification (not)* will be included in every configuration. For this reason, $notification$ can be regarded as *false optional*.

*Redundant constraint*. Constraint $c_9$ can be regarded as redundant (assuming that constraint $c_{14}$ has been deleted) since feature *participation* is part of every configuration (it is required by feature *decisionsupport*).

| property | property check | analysis operation |
|---|---|---|
| void feature model | inconsistent(FM) | $\Delta \subseteq FM : consistent(FM - \Delta)$ |
| test-induced void feature model | $\exists t_i \in T : inconsistent(FM \cup \{t_i\})$ | $\Delta \subseteq FM, \forall t_i \in T : consistent(FM - \Delta \cup \{t_i\})$ |
| dead feature $f_i$ | $inconsistent(\{f_i = true\} \cup FM)$ | $\Delta \subseteq FM : consistent(FM - \Delta \cup \{f_i = true\})$ |
| full mandatory feature $f_i$ | $inconsistent(\{f_i = false\} \cup FM)$ | $\Delta \subseteq FM : consistent(FM - \Delta \cup \{f_i = false\})$ |
| false optional feature $f_i$ | $inconsistent(\{f_i = false \wedge f_{i-1} = true\} \cup FM)$ | $\Delta \subseteq FM : consistent(FM - \Delta \cup \{f_i = false \wedge f_{i-1} = true\})$ |
| redundant constraints $c_i \in FM$ | $\exists c_i \in FM : FM - \{c_i\} \models c_i$ | $FM_{nr} \subseteq FM : \forall c_i \in FM_{nr} : FM_{nr} - \{c_i\} \not\models c_i$ |
| implicit feature groups $\{f_a, f_b\} \in FM$ | $\exists \{f_a, f_b\} \subseteq F : inconsistent(\{f_a = true \wedge f_b = false \vee f_a = false \wedge f_b = true\} \cup FM)$ | $\Delta \subseteq FM : consistent(FM - \Delta \cup \{f_a = true \wedge f_b = false \vee f_a = false \wedge f_b = true\})$ |

*Table 1: Summary of analysis operations for feature models. $FM_{nr}$ denotes a non-redundant feature model.*

## 6 Research Issues

There are a couple of open issues for future research directly related to the detection and resolution of inconsistencies in feature models.

*Personalized Conflict Detection and Resolution for Feature Modelling*. When developing feature models, inconsistencies in models can be resolved in different ways (in many cases, there exist different conflict sets). A task in this context is to propose conflict resolutions that are relevant, i.e., will be accepted by engineers. When developing knowledge bases, this means to figure out relevant faulty constraints where preferences can be derived from the development history of the feature model (e.g., how often a constraint has been changed in the last year, how often a constraint has been activated when determining a solution etc.). Initial related work can be found, for example, in (Felfernig et al. 2015).

*Anytime Conflict Detection*. There is a tradeoff between efficiency and accuracy that has to be taken into account when applying conflict detection algorithms (Felfernig et al. 2018). A challenge in this context is to identify conflicts of relevance for the user within time limits acceptable for interactive settings. Anytime conflict detection and diagnosis algorithms are an area of research that have to be investigated and further developed in the context of feature model development.

*Cognitive Effects*. An important question to answer is how knowledge engineers and domain experts without computer science background interpret the semantics of models (Fliedl et al. 2000; Michael and Mayr 2017). A related question is how easy it is for them to understand existing feature models and in which way we have to define natural language statements that represent constraints in feature models. In a typical feature model development process, constraints and feature hierarchies are defined by domain experts and then formalized by knowledge engineers. In this context, it has to be assured that knowledge engineers understand (textually defined) domain constraints to reduce the number of interaction cycles between domain experts and knowledge engineers (Fliedl et al. 2007; Kop and Mayr 1998; Shekhovtsov et al. 2014).

*Model Development Practices*. A major issue in the context of feature model development and maintenance is how to structure the domain knowledge in such a way that changing chunks of knowledge can be easily managed. Maintainability of variability models is still an open research issue and especially in the context of large and

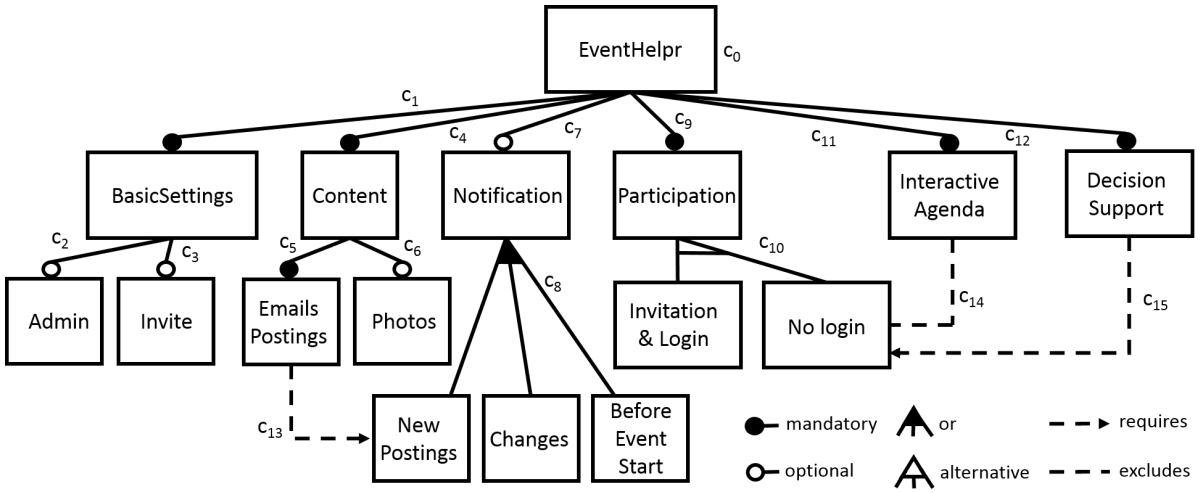



*Figure 2: Example of a faulty feature model. Dashed arrows specify* requires *and dashed lines* excludes *constraints.*

complex models, structuring mechanisms have to be developed that help to achieve the goal of easy maintenance. There is a branch of research dealing with collecting knowledge from the crowd and synthesizing configuration knowledge thereof (Ulz et al. 2016). This is also a promising area for the development of feature models.

*Merging of Feature Models.* In global contexts, for example, a car company selling cars in countries with different legal contexts, feature models have to take into account contextual information (sales strategies can differ and also country-specific legal aspects have to be taken into account). To analyze (e.g., in how many countries is it possible to sell feature $x$?) and adapt the underlying feature models, country-individual models have to be integrated. Existing research in the area of model integration (Liberatore and Schaerf 1998) has to be analysed with regard to applicability in the context of variability modelling.

## 7 Conclusions

Requirements Engineering (RE) is a critical phase in software development processes. In this article, we focused on software variability modelling scenarios where features are used to specify the inclusion or exclusion of specific software functionalities. With increasing size and complexity of those models, the probability of including inconsistencies increases. We provided an overview of different types of inconsistencies and showed how to automatically detect these inconsistencies on the basis of the concepts of conflict detection and model-based diagnosis.

## References

Bakker R., Dikker F., Tempelman F., Wogmim P. (1993) Diagnosing and Solving Over-determined Constraint Satisfaction Problems. In: 13th International Joint Conference on Artificial Intelligence. Chambery, France, pp. 276–281

Batory D. (2005) Feature Models, Grammars, and Propositional Formulas. In: Software Product Lines Conference. Springer Lecture Notes in Computer Science Vol. 3714, pp. 7–20

Benavides D., Felfernig A., Galindo J., Reinfrank F. (2013) Automated Analysis in Feature Modelling and Product Configuration. In: 13th International Conference on Software Reuse. Lecture Notes in Computer Science 7925. Springer, Pisa, Italy, pp. 160–175

Benavides D., Segura S., Ruiz-Cortes A. (2010) Automated Analysis of Feature Models 20 years Later: a Literature Review. In: Information Systems 35(6), pp. 615–636

Czarnecki K., Helsen S., Eisenecker U. (2005) Formalizing Cardinality-based Feature Models and their Specialization. In: SoftwareProcess: Improvement and Practice 10(1), pp. 7–29

Felfernig A., Friedrich G., Jannach D., Stumptner M. (2004) Consistency-based Diagnosis of Configuration Knowledge Bases. In: Artificial Intelligence 152(2), pp. 213–234

Felfernig A., Hotz L., Bagley C., Tiihonen J. (2014) Knowledge-based Configuration: From Research to Business Cases, 1st. Elsevier/Morgan Kaufmann

Felfernig A., Reiterer S., Stettinger M., Tiihonen J. (2015) Intelligent Techniques for Configuration Knowledge Evolution. In: Vamos 2015 Workshop. Hildesheim, Germany, pp. 51–60

Felfernig A., Walter R., Galindo J., Benavides D., Atas M., Polat-Erdeniz S., Reiterer S. (2018) Anytime Diagnosis for Reconfiguration. In: Journal of Intelligent Information Systems (JIIS)

Fliedl G., Kop C., Mayr H., Mayerthaler W., Winkler C. (2000) Linguistically based requirements engineering - The NIBA-project. In: Data & Knowledge Engineering 35 (2), pp. 111–120

Fliedl G., Kop C., Mayr H., Salbrechter A., Vöhringer J., Weber G., Winkler C. (2007) Deriving static and dynamic concepts from software requirements using sophisticated tagging. In: Data & Knowledge Engineering 61 (3), pp. 433–448

Junker U. (2004) QuickXPlain: Preferred Explanations and Relaxations for Over-constrained problems. In: 19th Intl. Conference on Artifical Intelligence (AAAI'04). AAAI Press, San Jose, California, pp. 167–172

Kang K., Cohen S., Hess J., Novak W., Peterson S. (1990) Feature-oriented Domain Analysis (FODA) – Feasibility Study. In: Technical Report, CMU-SEI-90-TR-21

Kop C., Mayr H. (1998) Conceptual predesign bridging the gap between requirements and conceptual design. In: 3rd International Conference on Requirements Engineering. Colorado Springs, CO, USA, pp. 90–98

Leffingwell D., Widrig D. (2003) Managing Software Requirements: A Use Case Approach, 2nd. Addison-Wesley

Liberatore P., Schaerf M. (1998) Arbitration (or how to merge knowledge bases). In: IEEE Transactions on Knowledge and Data Engineering 10 (1), pp. 76–90

Mayr H., Kop C. (2002) A User Centered Approach to Requirements Modeling. In: Modellierung 2002, pp. 75–86

Mayr H., Kop C., Esberger D. (2007) Business Process Modeling and Requirements Modeling. In: International Conference on the Digital Society (ICDS 2007). Guadeloupe, French Caribbean, p. 8

Michael J., Mayr H. (2017) Intuitive understanding of a modeling language. In: Australasian Computer Science Week Multiconference (ACSW 2017). Geelong, Australia, 35:1–35:10

Reiter R. (1987) A Theory of Diagnosis From First Principles. In: Artificial Intelligence 32(1), pp. 57–95

Shekhovtsov V., Mayr H., Kop C. (2014) Facilitating effective stakeholder communication in software development processes. In: Forum at the Conference on Advanced Information Systems Engineering (CAiSE), pp. 116–132

Thum T., Batory D., Kastne C. (2009) Reasoning about edits to feature models. In: 31st IEEE International Conference on Software Engineering. IEEE, pp. 254–264

Tsang E. (1993) Foundations of Constraint Satisfaction. Academic Press, London, San Diego, New York

Ulz T., Schwarz M., Felfernig A., Haas S., Reiterer S., Stettinger M. (2016) Human Computation for Constraint-based Recommenders. In: Journal of Intelligent Information Systems (JIIS)

White J., Benavides D., Schmidt D., Trinidad P., Dougherty B., Ruiz-Cortez A. (2010) Automated Diagnosis of Feature Model Configurations. In: Systems and Software 83(7), pp. 1094–1107

# Conceptual Modelling of Service-Oriented Software Systems

Klaus-Dieter Schewe[*,a], Karoly Bósa[a], Andreea Buga[a], Sorana Tania Nemeş[a]

[a] Christian-Doppler Laboratory for Client-Centric Cloud Computing, Hagenberg, Austria

Abstract. *Conceptual modelling has originated from the areas of software engineering, databases and knowledge representation, and Heinrich C. Mayr, to whom this article is dedicated, has been involved in this area from the very beginnings. While in these areas a high degree of maturity has been achieved, conceptual modelling still lacks this maturity in other areas such as service-oriented systems despite the demand from novel application areas such as cloud computing. In this article we discuss the axiomatic BDCM[2] framework capturing **b**ehaviour, **d**escription, **c**ontracting, **m**onitoring and **m**ediation. We argue that the framework gives an abstract answer to the ontological question what service-oriented systems are. On these grounds we address the intrinsically connected modelling question how to capture cloud-enabled service-oriented systems. We outline a conceptual modelling approach that is grounded in a distributed middleware coordinating the client access to multiple clouds through a concept of mediator. For this we exploit abstract machines with interconnected layers for normal operation, monitoring and adaptation. We illustrate the model by the use case of a robotic care system showing that the general model can be fruitfully exploited for failure alerts, failure anticipation and prevention, and safety hazards detection, which links the research to recent interests of Heinrich in conceptual modelling for ambient assistance systems.*

Keywords. Conceptual Modelling • Service-Oriented System • Ambient Assistance

## 1 Introduction

The field of conceptual modelling has originated from the areas of software engineering, databases and knowledge representation (see e. g. the early collection by Brodie 1984). The emphasis was to create a bridge connecting the informal understanding of stakeholders and the formally precise understanding of system developers. The conceptual model should be a precise blueprint of a system to be built, which at the same time is on a sufficiently high level of abstraction to support mutual comprehensibility. Conceptual modelling was successfully applied to the development of data-centric information systems. The so-called semantic data models (see Hull and King 1987 for a survey) played an important role in this success, in particular the Entity-Relationship model (extended to perfection by Thalheim 2000) was adopted by researchers, consultant, users and developers.

In the further development of the field two main directions of the research can be identified. The first direction is dedicated to novel modelling methods that extend the mature ones in ways that enable different classes of systems to be addressed. Prominent examples are web information systems by Schewe and Thalheim (2018) and business processes by Kossak et al. (2016). The other direction emphasises the application of the methods in specialised application areas such as e-commerce (see e. g. Kaschek et al. 2006) or e-learning (see e. g. Schewe et al. 2005).

Heinrich C. Mayr, to whom this article is dedicated, has been an active researcher in the area of conceptual modelling from the various beginnings.

* Corresponding author.
E-mail. kdschewe@acm.org
The research reported in this paper has been supported by the Christian-Doppler Society in the frame of the Christian-Doppler Laboratory for Client-Centric Cloud Computing.

In particular, he emphasised the generation of skeletons of conceptual database schemata by applying natural language processing. Fliedl et al. (2005) summarise main results of this research. This is also connected to research on the conceptual modelling process, in particular the involvement of stakeholders addressed by Shekhovtsov et al. (2012). While this research stays with the original emphasis of conceptual modelling in the context of the development of data-centric information systems, his research also addressed ontologies in conceptual modelling (see Hesse et al. 2004) and business processes (see Mayr et al. 2007).

His more recent research has been directed towards specific applications. Together with colleagues (see Michael and Mayr 2016) he investigated domain-specific conceptual modelling in general, and in particular the application of conceptual modelling to ambient assistance systems (see among others Al Machot et al. 2014 and Michael and Mayr 2013).

### 1.1 Service-Oriented Systems

Surprisingly, despite the long successful history of conceptual modelling, there is no commonly accepted conceptual model for service-oriented systems, not even an understanding what services are. According to (Bergholtz et al. 2015) the many approaches to service-oriented systems can be roughly classified into those taking a business-centric view and those taking a software-centric view. Examples of the former class are among others the service science framework by Ferrario et al. (2011), the so-called Unified Theory of Services (UTS) by Sampson and Froehle (2006), the approach to semantic web services by Preist (2004) or the OASIS framework by Alves et al. (2007). Examples of the latter class are among others the service-oriented architecture frameworks (SOA) by Arsanjani et al. (2008), Erl (2007) and Papazoglou and van den Heuvel (2006), service-oriented computing (SOC) by Papazoglou and van den Heuvel (2007), web services by Alonso et al. (2003) and Benatallah et al. (2006) and the corresponding W3C standards (Universal Description, Discovery and Integration (UDDI) 2009 and Simple Object Access Protocol (SOAP) 2008), and the behavioural model of Abstract State Services by Ma et al. (2008) and its extension by Ma et al. (2009a). Both lists can be extended by many other examples (see for instance the literature review by Ma et al. 2009a or by Bergholtz et al. 2015).

The research towards the fundamental question "what constitutes a service" has led to the model of Abstract State Services ($AS^2$s) (see Ma et al. 2009a), the model of service mediators based on $AS^2$s (see Ma et al. 2012 and Schewe and Wang 2012), and the service ontology model (see Ma et al. 2009b). The discussion of further properties characterising a software service (on the web) led to the $BDCM^2$ framework capturing **b**ehaviour, **d**escription, **c**ontracting, **m**onitoring and **m**ediation (see Schewe and Wang 2015). In addition, the W*H framework by Dahanayake and Thalheim (2015) puts services into the context of their usage, intention and added value, which are more relevant for the modelling of service-oriented systems than for the clarification of the question what they are.

The $BDCM^2$ framework has been successfully adopted in a conceptual model for multi-cloud interaction by Buga et al. (2017a). The emphasis is on distributed systems exploiting software services from multiple clouds. The services themselves can be modelled as abstract state services specified by Abstract State Machines (ASMs) (see Börger and Stärk 2003 for a general introduction to ASMs). The services can be integrated using the mediator model, by means of which general skeletons are provided that are instantiated by concrete services that are selected according to a service ontology comprising functional, categorical and SLA-based properties. The concrete interaction of a mediator instance with the service providing clouds is subject of a middleware system, developed by Bósa et al. (2015) (see also Bósa 2012 and Bósa et al. 2014), which handles the interaction with the clouds and supports the interaction between different systems through the clouds (see Bósa 2013 for this aspect). For the rigorous specification of this middleware the ambient extension of ASMs by Börger et al. (2012) has been exploited.

The monitoring aspect with respect to security and SLA satisfaction has been addressed by Lampesberger and Rady (2015). Buga et al. (2017b) show how such cloud-enabled distributed systems can be used for modelling a robotic care system showing that the general model can be fruitfully exploited for failure alerts, failure anticipation and prevention, and safety hazards detection. This links the research to recent interests of Heinrich in conceptual modelling for ambient assistance systems.

### 1.2 Organisation of the Article

In Section 2 we address the ontological question: what are services and what are service-oriented systems. Here we refer to the BDCM$^2$ framework and its underpinnings in the behavioural theory of distributed adaptive systems. That is, we emphasise an axiomatic characterisation, though we will abstain from going too much into theoretical depth. Section 3 is then dedicated to the intrinsically connected modelling question: how can service-oriented systems be modelled conceptually. Here our research emphasises Abstract State Machines with various extensions as well as service ontologies as the key building blocks. In view of the many non-technical additions that are needed in design and development of very complex software systems (as discussed by Dahanayake and Thalheim 2015 for services and deeply investigated by Schewe and Thalheim (2018) for web information systems that are largely related) we acknowledge that our presentation here does not cover the complete picture. To illustrate the model we briefly outline the application to the robotic care system use case. Finally, in Section 4 we discuss perspectives for future research in this area, which are an open invitation to Heinrich and his colleagues to remain active and to further contribute to the area of conceptual modelling.

## 2 The Ontological Question: What *Are* Services and Service-Oriented Systems?

The BDCM$^2$ framework addresses the question how to obtain a language independent definition of the notions of service and service-oriented system. We will rephrase the definition in an axiomatic way, but for the sake of brevity we have to leave out many technical details, for which we refer to the literature.

### 2.1 Functional Behaviour

A *behavioural theory* provides first a language-independent clarification of a class of systems by means of a set of axioms, which is complemented in a second step by an abstract machine model, for which it is then proven in a third step that the machine model captures exactly the systems stipulated by the axioms. For the case of services, Ma et al. (2009a) started with the introduction of Abstract State Service (AS$^2$) defining the functional behaviour of a service. Roughly speaking, an AS$^2$ provides a process that can be used by someone else knowing only what the process implementing the service is supposed to do. The user does neither own the service nor is he able to manipulate it.

Each service will provide some form of workflow that will access data resources. Therefore, the AS$^2$ model refers to an underlying database (using this term in a very general sense), which defines an internal layer. For services there must be an additional external layer made out of views, which export the data that can then be used by users or programs. We complete this picture by adding operations on both the conceptual and the external layer, the former one being handled as database transactions, whereas the latter ones provide the means with which users can interact with a database.

**Axiom 1.** A service comprises an internal (database) layer, which consists of

- a set $\mathcal{S}$ of states together with a subset $\mathcal{I} \subseteq \mathcal{S}$ of initial states,
- a wide-step transition relation $\tau \subseteq \mathcal{S} \times \mathcal{S}$, and
- a set $\mathcal{T}$ of transactions, each of which is associated with a small-step transition relation $\tau_t \subseteq \mathcal{S} \times \mathcal{S}$ ($t \in \mathcal{T}$) satisfying the axioms of a database transformation over $\mathcal{S}$.

**Axiom 2.** A service comprises an external (view) layer comprising a finite set $\mathcal{V}$ of (extended) views.

- Each view $v \in \mathcal{V}$ is associated with a database transformation such that for each state $S \in \mathcal{S}$ there are views $v_1, \ldots, v_k \in \mathcal{V}$ with finite runs $S_0^j, \ldots, S_{n_j}^j$ of $v_j$ $(j = 1, \ldots, k)$, starting with $S_0^j = S_d$ and terminating with $S_{n_j}^j = S_d \cup V_j$.
- Each view $v \in \mathcal{V}$ is further associated with a finite set $O_v$ of (service) operations $o_1, \ldots, o_n$ such that for each $i \in \{1, \ldots, n\}$ and each $S \in \mathcal{S}$ there is a unique state $S' \in \mathcal{S}$ with $(S, S') \in \tau$.
- If $S = S_d \cup V_1 \cup \cdots \cup V_k$ with $V_i$ defined by $v_i$ and $o$ is an operation associated with $v_k$, then $S' = S'_d \cup V'_1 \cup \cdots \cup V'_m$ with $m \geq k - 1$, and $V'_i$ for $1 \leq i \leq k - 1$ is still defined by $v_i$.

We omit further technical details of the model of abstract state services (for these see Ma et al. 2009a or Schewe and Wang 2015). Both axioms above refer to database transformations, which have been axiomatically defined by Schewe and Wang (2010) with a further sharpening of the underlying behavioural theory by Ferrarotti et al. (2016). Database transformation can be axiomatically characterised by four further axioms, which we will sketch next (for details we have to refer to the lengthy treatment in the literature).

**Axiom 3.** A database transformation comprises a set $\mathcal{S}$ of states, a subset $\mathcal{I} \subseteq \mathcal{S}$ of initial states, and a state transition function $\varsigma \subseteq \mathcal{S} \times \mathcal{S}$ (sequential time).

**Axiom 4.** States of a database transformation are meta-finite states over a signature $\Sigma = \Sigma_{db} \cup \Sigma_a \cup \Sigma_b$ comprising database functions, algorithmic functions and bridge functions. States are closed are isomorphisms (abstract state).

**Axiom 5.** A database transformation comprises a background structure, which captures at least truth values, constructors for records and finite multisets, and eventually further constructors for the building blocks of the database model as well as operators associated with these constructors for values and terms (background).

**Axiom 6.** There is a finite set $W$ of multiset comprehensions terms built over $\Sigma$ and the background structure such that for any two states $S_1$, $S_2$ that coincide on them the corresponding update sets $\Delta(S_1)$ and $\Delta(S_2)$ (defined by the state transition function) are equal (bounded exploration).

## 2.2 Description of Services

We may assume that abstract state services as defined above are available through service repositories, for which we adopt the notion of "cloud". In a service-oriented system several such services are combined including third-party services. For the latter ones it has to be known, which functionality is provided, despite the fact that the internal layer (and thus the implementation) is hidden. So the question arises how meaningful services can be located, for which a description of the available services is needed. The key idea is that given a coarse description of the service needed, it should be possible to search for such a service. This is exactly what ontologies are meant to capture, a description of the (semantics of) services. Therefore, we adopt the common idea of a service ontology, which is already omnipresent in the area of the semantic web, also in our own work (see e. g. Ma et al. 2012, 2009b).

This is usually grounded in some more or less expressive description logic, e.g. the web ontology language OWL (see Grau et al. 2008). While it is not possible to precisely describe an abstract state service in a way that it can be automatically matched to a service request, it is commonly accepted that a service ontology should capture three aspects (see e. g. Fensel et al. 2007 and the references in there), which leads to the following axiom:

**Axiom 7.** Each service is equipped with a description in a service ontology, which comprises

- a *functional* description covering the specification of input- and output types as well as

pre- and post-conditions telling in technical terms, what the service will do;

- a *categorical* description capturing interrelated keywords telling what the service operation does by using common terminology of the application area;
- a *quality of service* (QoS) description capturing non-functional properties such as availability, response time, cost, etc.

A functional description alone would be insufficient. For instance, a flight booking service may functionally be almost identical to a train booking system. Therefore, an additional categorical description is indispensable. The terminology of the application domain defines an ontology in the widest sense, i. e.. we have to provide definitions of "concepts" and relationships between them, such that each offered service becomes an instantiation of one or several concepts in the terminology.

The QoS description is not needed for service discovery, but for selection among alternatives. However, the non-functional properties cover also the description of how a service is meant to be used, what are the obligations and rights of the participating agents, at least of the service provider and the service user, how conflicts are to be handled, etc. These non-functional aspects should be considered as being intrinsically part of a service. (Schewe and Wang 2015) discuss them in detail.

## 2.3 Service Mediation

Service mediation addresses collaboration of services, which is another necessary aspect of services. While the functional behaviour of a service is entirely in the hands of a service provider and the service description addresses how a service is offered by the provider to the potential users, service mediation covers how users can make use of a service. For instance, it is commonly known that online sales services are often used as information repositories without the slightest intention to buy something. In other words, while the functionality of a service is defined by the provider and conditions of use are subject of the QoS part of the service ontology, it is exclusively up to the user to exploit the service for his own purposes. Therefore, service mediation is an important aspect of a theory of services.

First, in order to make the workflow within a service explicit we require the notion of *plot*, which actually captures the possible sequencing of service operations. For this Ma et al. (2012) exploit Kleene algebras with tests (KAT), which are known to be the most expressive formalism to capture propositional process specifications. So adding plots to the $AS^2$ model is a little extension of the behavioural model, which in addition impacts on the functional description in the service ontology.

Formally, the service operations give rise to *elementary processes* of the form

$$\varphi(\vec{x})\ op[\vec{z}](\vec{y})\ \psi(\vec{x},\vec{y},\vec{z}),$$

in which $op$ is the name of a service operation, $\vec{z}$ denotes input for $op$ selected from the view $v$ with $op \in Op_v$, $\vec{y}$ denotes additional input from the user, and $\varphi$ and $\psi$ are first-order formulae denoting pre- and postconditions, respectively. The pre- and postconditions can be omitted; also simple formulae $\chi(\vec{x})$ interpreted as tests checking their validity constitute elementary processes. With this (Ma et al. 2012) define the set of *process expressions* of an $AS^2$ as the smallest set $\mathcal{P}$ containing all elementary processes that is closed under sequential composition $\cdot$, parallel composition $\|$, choice $+$, and iteration $^*$. That is, whenever $p, q \in \mathcal{P}$ hold, then also $pq$, $p\|q$, $p+q$ and $p^*$ are process expressions in $\mathcal{P}$. However, this definition is tightly linked to KATs; it can be generalised in a language-independent way emphasising any kind of parallel process (as stipulated by Axioms 3-6) that are composed out of the given elementary processes.

**Axiom 8.** Each service is equipped with a *plot*, which is a process expression over the service operations of the service.

Second, service mediators exploit plots with open slots for services to specify intended service-based applications on a high level of abstraction. The idea is to specify service-oriented applications that involve yet unknown component services. On these grounds matching criteria for services are formally defined that are to fill the slots. A problem in finding such matching criteria is the fact that it should be possible to skip component operations of services and change their order. This enhances the work on service composition, which is already a well-explored area in service computing with respect to services that are understood functionally. In the $AS^2$ model this corresponds to the service operations rather than the services as a whole. More precisely, what actually needs to be composed are "runs" of services that are determined by the plot including conditions, under which particular service operations can be removed or their order can be changed. This leads to rather complicated matching conditions between services and slots in mediators as emphasised by Ma et al. (2012) and further refined by Schewe and Wang (2012).

In order to capture this idea we relax the definition of a plot in such a way that service operations do not have to come from the same service. Thus, in elementary processes we use prefixes to indicate the corresponding $AS^2$, so we obtain $\varphi(\vec{x})\ X : op[\vec{z}](\vec{y})\ \psi(\vec{x},\vec{y},\vec{z})$, in which $X$ denotes a *service slot*, i. e. a placeholder for an actual service.

**Axiom 9.** A service-oriented system consists of service mediators, which are process expressions with service slots. Each service operation in a mediator is associated with input- and output-types, pre- and postconditions, and a concept in a service terminology. An instance of a mediator is defined by services matching the slots.

A service mediator specifies, which services are needed and how they are composed into a new plot of a composed $AS^2$. So we now need exact criteria to decide, when a service matches a service slot in a service mediator.

It seems rather obvious that in such matching criteria for all service operations in a mediator associated with a slot $X$ we must find matching service operations in the same $AS^2$, and the matching of service operations has to be based on their functional and categorical description. Functionally, this means that the input for the service operation as defined by the mediator must be accepted by the matching service operation, while the output of the matching service operation must be suitable to continue with other operations as defined by the mediator. This implies that we need supertypes and subtypes of the specified input- and output-types, respectively, in the mediator, as well as a weakening of the precondition and a strengthening of the postcondition. Categorically, the matching service operation must satisfy all the properties of the concept in the terminology that is associated with the placeholder operation, i. e.. the concept associated with the matching service operation must be subsumed by that concept. We also have to ensure that the projection of the mediator to a particular slot $X$ results in a subplot of the plot of the matching $AS^2$. The order of service operations may differ and certain service operations may become redundant, which has to be taken care of as well. (Ma et al. 2012) discuss matching criteria in detail.

## 2.4 Service Contracts

The aspects of behaviour, description and mediation alone do not yet capture everything that would characterise services. Zeithaml et al. (1985) emphasise general properties such as intangibility, inseparability, heterogeneity and perishability of services, which in the community have been controversially debated and rejected as being neither necessary nor sufficient. Indeed, intangibility refers to the fact that a service is owned by its provider, while a service user can exploit the service for his own purpose, but there is never a transfer of ownership nor does the user even know how the service is realised. For instance, in the often used example of a snow removal service it is up to the provider to decide whether shovels, brooms or snow ploughs are used, which the service client is

not interested in at all, as long as the agreed result is guaranteed. Intangibility is de facto captured by the behavioural model; in particular, in the $AS^2$ model a hidden internal layer is separated from the visible layer exposed to the user, which includes the associated plot. Similarly, perishability is no property of services at all. Every software system is prone to perishability, if the hardware and system software it is grounded in disappears. To the contrary, it should be part of the usage agreement between a service provider and a user that services do not perish within the agreed period of service usage. That is, availability agreements are part of the service as well as the consequences, if the agreement is violated. Inseparability has to be treated also with care. Of course, from the point of view of the provider the service is offered as a whole, and there is an agreement about this with each service user. However, as already discussed in connection with mediation, it is up to the user to exploit the services in his own way and for his own purposes, regardless what the provider intended. In this sense, a user may well cut out of a service the parts that are really needed, even though for the provider it still appears that the service was used as a whole. Finally, heterogeneity is not a characteristic at all, as it could only refer to the collection of all services, in which case it is a triviality. If it were to refer to a single service, there is no reason for the claim that heterogeneity of the components involved should always be the case.

Schewe and Wang (2015) discuss in detail alternative ideas by Bergholtz et al. (2015), in the UTS by Sampson and Froehle (2006), in the REA ontology, and in the classification of rights following Hohfeld. While we cannot repeat this intensive discussion here, we emphasise the conclusion that this would still draw an incomplete picture regarding non-functional aspects, as there are more rules than those capturing rights and obligations. In general, there should be a whole list of service-level agreements, which altogether define a contract between the provider of a service and a service user. Rady (2012) has defined fragments of an ontology capturing SLAs and implemented a tool extracting contracts from such an SLA ontology (see Rady 2014). For the time being the SLA ontology, which in our opinion should be part of the service ontology, only addresses availability and performance aspects, though more than 20 additional types of SLAs have already been discussed in the literature. SLAs concerning availability, performance, etc. but also security and privacy regulations have nothing to do with rights. For instance, availability on one hand concerns conditions of usage, e.g., if the service is only available on workdays within a specified time period, and on the other hand a commitment and obligation by the provider. Security and privacy may be also handled as commitments, but it should preferably also include the means with which security is supposed to be achieved, in which case the SLA includes a statement about the functional behaviour of the service or its environment. In summary, for each service there must exist a service contract capturing all SLAs, and the SLAs may be expressed by obligations and rights in a deontic action logic, refer to functional aspects, or simply cover factual data. The decisive feature, however, is that a model of services must comprise the ***c****ontracting* aspect.

**Axiom 10.** For each service used in a service-oriented system there must exist a contract between service provider and service user covering all relevant SLAs for the service. The conditions in the contract must be consistent with the quality of service description in the service ontology.

## 2.5 Monitoring and Adaptation

(Schewe and Wang 2015) emphasise that a contract that cannot be validated is rather useless, both technically and legally. Therefore, for every service it should be possible to check not only its behaviour, but also whether the SLAs are fulfilled. That is, there must exist a monitoring software for this purpose – this could again be a service, but not necessarily. Lampesberger and Rady (2015) make a promising step in the direction of SLA monitoring in the context of cloud computing.

While monitoring is a decisive aspect of service-oriented systems, the focus on SLA monitoring is too narrow.

Lampesberger (2016) already emphasises anomaly detection as another aspect, for which monitoring is required, in particular, if services exploit public clouds or are in any other way exposed to undesired external interference. This naturally extends to failures and any other critical situation as emphasised by Buga et al. (2017a).

**Axiom 11.** Every instance of a service-oriented system constitutes a distributed adaptive system, in which all transactions and service operations of individual services are reflective, i. e.. they satisfy the axioms for reflective database transformations.

Note that this axiom includes the possibility to provide specialised services for monitoring (as e. g. for anomaly detection) and for adaptation, e. g. replacing services in an instantiation as discussed by Buga et al. (2017a). For the extension, that reflective behaviour has to be supported; we refer to the work by Schewe et al. (2017), who discuss the axioms for distributed adaptive systems in detail.

For our brief exposition here we restrict to mention that only axioms 3-6 need to be slightly amended. In a nutshell, reflection can be captured by storing the specification of the system behaviour, e.g. the signature and ASM rule as part of the system's states. Then terms over the signature $\Sigma$ can also be used as values, and the interpretation of some terms in a state may result in terms. For the bounded exploration axiom we then need a stronger notion for the coincidence of terms that are extended in this way. For strong coincidence we request (as before) that the interpretation of the terms in $W$ is the same for two states, but in addition the interpretation of terms resulting from this first interpretation also yields equal results. Axiom 6 can then be rephrased as follows:

**Axiom 6′.** There is a finite set $W$ of multiset comprehensions terms built over $\Sigma$ and the background structure such that for any two states $S_1$, $S_2$ that strongly coincide on them the corresponding update sets $\Delta(S_1)$ and $\Delta(S_2)$ are equal (reflective bounded exploration).

## 3 The Modelling Question: How to *Capture* Service-Oriented Systems?

Our axiomatic definition of service-oriented systems is language-independent, so for the actual task to model service-oriented systems we require adequate languages by means of which we can obtain models satisfying the axioms. Actually, in behavioural theories we go further asking for the capture of the class of systems of interest, which requires to formally prove that all systems as stipulated by the axioms are faithfully represented by the modelling language. For conceptual modelling the focus is in addition on the capture on a high level of abstraction, by means of which the comprehensibility requirement can be fulfilled.

With respect to the axioms describing functional behaviour, mediation and monitoring the concepts of abstract state services, mediators and reflection together, i. e. all axioms except Axioms 7 and 10, give rise to distributed adaptive systems. In a longer sequence of theoretical research on foundations of such systems a behavioural theory was discovered. In a nutshell, the final result is that distributed adaptive systems are captured by concurrent reflective ASMs. Here we abstain from a presentation (for technical details see Schewe et al. 2017 and the references in there). Instead we concentrate on a cloud-based middleware architecture that enables distributed adaptive service-oriented systems. While the middleware itself is specified and verified using concurrent reflective ASMs, applications can exploit the $AS^2$ model for modelling of services, mediators for creating service-oriented systems, and an associated service ontology.

### 3.1 Cloud-Enabled Service-Oriented Systems

In the previous section we emphasised that services should be taken from a service repository. We deliberately use the term *cloud* to refer to such

a repository of services as emphasised by Ma et al. (2012). From the definition of mediators it is clear that an instantiated mediator is a high-level specification of a distributed application that runs several services at the same time. Refining and implementing such a specification requires several add-ons. The involved services have to be started and terminated, which usually involves a log-in and authentication process. Then data has to be passed from the mediation process to the individual services, which bypass the user interaction, i. e. a control component associated with the process is needed. Furthermore, output from several services is combined, and a selection made by a user is passed back to the originating services, while non-selection leads to service termination. This must also be handled by the control component, for which we employ the *client-cloud interaction middleware* (CCIM) model defined by Bósa et al. (2014).

**Client-Cloud Interaction Middleware**

The CCIM has been specified using ambient ASMs in order to describe formal models of distributed systems incorporating mobile components in two abstraction layers. While the algorithms of executable components are specified in terms of ASMs, their communication topology, locality and mobility are described with the terms of ambient calculus. As each ambient ASM specification can be translated into a pure ASM specification as shown by Börger et al. (2012). The approach provides a universal way to handle client-cloud interaction independent from particularities of certain cloud services or end-devices, while the instantiation by means of particular ambient results in specifications for particular settings. Thus, the architecture is highly flexible with respect to additional end-devices or cloud services, which would just require the definition of a particular ambient. The architecture of the CCIM integrates all novel software solutions such as Service Plot-Based Access Management, Client-to-Client Interaction (CTCI) Feature, Identity and Access Management (IdMM), Content Adaptivity, SLA Management and Security Monitoring Component into a compound single software component.

In the general architecture (see e. g. Buga et al. 2017a) the middleware is replicated by several components, each connected to one or more service clouds, but each cloud is connected to exactly one middleware component. Thus, there are three modes of interaction: (1) interaction of users with a middleware component, (2) interaction of a middleware component with a service in one of its clouds, and (3) interaction among several middleware components. The challenge is to keep users oblivious about the interaction among middleware components to locate individual services and to manage the transfer of results among the participating services. This challenge is addressed by the propagation of service requests among the middleware components. That is, when a middleware component receives a request for access to a particular service from a client or another middleware component, it will route the request to the middleware component owning the service, i. e. being the component that connects to the cloud, on which the service resides. Thus a service requests always comprises also routing information.

In addition to the routing of requests to access individual services each middleware component will exploit the features of the mediator model and analyse how to execute a particular mediator by extracting services that it can handle itself and those parts that have to be forwarded to other components. This is captured by an ambient-ASM specification of the distributed middleware emphasising the normal execution model. The normal execution mode requires the abstract machine, i. e. the ambientASM specification, the service interfaces, the request handler that links to the users and other middleware components, and the communication handler that handles the interaction among middleware components.

The CCIM provides a cloud service infrastructure that permits a transparent and uniform way for clients to interact with multiple clouds. It permits to access and combine the available functions of cloud services, which may belong to various owners, and it leaves the full control over the usage of

their services in the hands of the service owners. If a registered cloud user intends to subscribe to a particular service, a subscription request is sent to the cloud, which may forward it to such a special client corresponding to the service owner. This client responds with the service plot, which defines how the service can be used by the user and determines the permitted combination of service operations.

The received service plots are collected together with other available cloud functions in a personal user area by the cloud. When the subscribed user sends a service request, it is checked whether the requested service operations are permitted by any service plot. If a requested operation is permitted, then it is triggered to perform, otherwise it is blocked as long as a plot may allow to trigger it in the future. Each triggered operation request is authorized to enter into the user area of the corresponding service owner to whom the requested service operation belongs. Here a scheduler mechanism assigns to the request a one-off access to a cloud resource on which an instance of the corresponding service runs. Then the service operation request is forwarded to this resource, where the request is processed by an instance of the service whose operation was requested. Finally, the outcome of the performed operation returns to the area of the initiator user, where the outcome is either stored or send further to a given client device.

In this way, the service owners have direct influence on the service usage of particular users via the provided service plots. If a user subscribes to more than one service, he or she may have access to more than one plot. These plots are independent from each other and they can be applied concurrently. If a service owner makes available more than one service for a user, the owner has the choice either to provide independent plots for the user or to combine some functions of various services into a common service plot. This conceptual solution shows a transparent and uniform way how to provide an advanced access control mechanism for cloud services without giving up the flexibility of heterogeneous cloud access to these services.

Furthermore, due to the ambient concept the relocation of system components is trivial, and the model can be applied to different scenarios. For instance, all our novel methods including our client-cloud interaction solution can be shifted to the client side and wrapped into a middleware software which takes place between the end users and cloud in order to control the interactions of them. Note that the specified communication among the distributed system components remains the same in both scenarios.

**Monitoring and Adaptation**

In addition, the CCIM provides monitoring and assessment layers. For each service there are several dedicated monitors. For the observation of the behaviour of these services sensors are deployed across multiple clouds in order to collect environmental data that are reported to the middleware. The monitoring is part of an abstract machine, which is specified using the ASM method. It is important to consider that monitors are also components of the distributed system, so they can also exhibit failures themselves. This is taken care of by assigning a trustworthiness measure to each monitor. Monitoring components with trustworthiness below a specified threshold are removed from the network of monitors. In case of an identified critical situation the adaptation layer replaces one or several services, i. e. replacing the given mediator instantiation by a new one.

Monitors collect data from the nodes. When starting the system, each monitor is initialized by the middleware in the `Active` state, from where it submits a heartbeat request to the node it monitors. The monitor advances afterwards to the `Wait for response` state, where it checks two guards. First, it verifies if a response to its request is received. If so, it verifies if the delay of the response is acceptable. If this condition holds, the monitor moves to the `Collect data` state. If no response is received or if the response has a big delay, the monitor moves to the `Report problem` state. In the `Collect data` state the monitor gathers

low level metrics (CPU, memory and storage usage, bandwidth) and then moves to the `Retrieve information` state, where it checks local storage for past monitoring data. If the repository is available, the monitor queries it. The monitor moves to the `Assign diagnosis` state, where it interprets the available data. If it discovers a problem, it moves to the `Report problem` state, otherwise it moves to the `Log data` state, where meaningful data and operation are logged. When an issue is identified, the monitor modifies a constraint that triggers a request towards the leader of the node to inquire all its monitoring counterparts and carry out a collaborative diagnosis. After reporting the issue, the monitor moves to the `Log data` state. Here, the confidence degree of the monitor is checked, and if the monitor is still trustworthy, it starts a new monitoring cycle. Alternatively, it moves to the `Inactive` state and waits to be deployed again in the system.

Details of the CCIM model are covered in several articles, e. g. by Bósa et al. (2015), Buga et al. (2017a) and Lampesberger and Rady (2015). We omit further details here.

## 3.2 Service Ontology

As outlined, the functional, categorical and QoS description of services in a cloud requires the definition of an ontology. That is, we need a *terminological knowledge* layer (aka TBox in description logics) describing concepts and roles (or relationships) among them. This usually includes a subsumption hierarchy among concepts (and maybe also roles), and cardinality constraints. In addition, there is an *assertional knowledge* layer (aka ABox in description logics) describing individuals. Thus, services in a cloud constitute the ABox of an ontology, while the cloud itself is defined by the TBox (for details see Ma et al. 2012 and Schewe and Wang 2015).

### Terminologies

For simplicity let us assume that $C_0$ and $R_0$ represent not further specified sets of basic concepts and roles, respectively. Then *concepts* $C$ and *roles* $R$ are defined by the following grammar:

$$R = R_0 \mid R_0^-$$
$$A = C_0 \mid \top \mid \geq m.R \text{ (with } m > 0)$$
$$C = A \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists R.C \mid \forall R.C$$

A *terminology* (or TBox) is a finite set $\mathcal{T}$ of assertions of the form $C_1 \sqsubseteq C_2$ with concepts $C_1$ and $C_2$ as defined by the grammar above. Each assertion $C_1 \sqsubseteq C_2$ in a terminology $\mathcal{T}$ is called a *subsumption axiom*. As usual, we use the shortcut $C_1 \equiv C_2$ instead of $C_1 \sqsubseteq C_2 \sqsubseteq C_1$. For concepts, $\bot$ is a shortcut for $\neg\top$, and $\leq m.R$ is a shortcut for $\neg \geq m+1.R$.

A *structure* $\mathcal{S}$ for a terminology $\mathcal{T}$ consists of a non-empty set $O$ together with subsets $\mathcal{S}(C_0) \subseteq O$ and $\mathcal{S}(R_0) \subseteq O \times O$ for all basic concepts $R_0$ and basic roles $R_0$, respectively. $O$ is called the base set of the structure.

We first extend the interpretation of basic concepts and roles and to all concepts and roles as defined by the grammar above, i. e. for each concept $C$ we define a subset $\mathcal{S}(C) \subseteq O$, and for each role $R$ we define a subset $\mathcal{S}(R) \subseteq O \times O$ as follows:

$$\mathcal{S}(R_0^-) = \{(y,x) \mid (x,y) \in \mathcal{S}(R_0)\}$$
$$\mathcal{S}(\top) = O$$
$$\mathcal{S}(\neg C) = O - \mathcal{S}(C)$$
$$\mathcal{S}(\geq m.R) = \{x \in O \mid \#\{y \mid (x,y) \in \mathcal{S}(R)\} \geq m\}$$
$$\mathcal{S}(C_1 \sqcap C_2) = \mathcal{S}(C_1) \cap \mathcal{S}(C_2)$$
$$\mathcal{S}(C_1 \sqcup C_2) = \mathcal{S}(C_1) \cup \mathcal{S}(C_2)$$
$$\mathcal{S}(\exists R.C) = \{x \in O \mid (x,y) \in \mathcal{S}(R) \text{ for some } y \in \mathcal{S}(C)\}$$
$$\mathcal{S}(\forall R.C) = \{x \in O \mid (x,y) \in \mathcal{S}(R) \Rightarrow y \in \mathcal{S}(C) \text{ for all } y\}$$

A *model* for a terminology $\mathcal{T}$ is a structure $\mathcal{S}$, such that $\mathcal{S}(C_1) \subseteq \mathcal{S}(C_2)$ holds for all assertions $C_1 \sqsubseteq C_2$ in $\mathcal{T}$. A finite model, i. e. a model with a finite base set, is also called *instance* or ABox associated with $\mathcal{T}$.

**Functional and Categorical Description**
As outlined above we expect the terminology $\mathcal{T}$ of a cloud to provide the functional, categorical and QoS description of its offered services. The functional description of a service operation consists of input- and output-types, and pre- and post-conditions. For the types we need a type system with base types and constructors. For instance, the following grammar

$$t = b \mid \mathbb{1} \mid (a_1 : t_1, \ldots, a_n : t_n) \mid \{t\} \mid [t] \mid (a_1 : t_1) \oplus \cdots \oplus (a_n : t_n)$$

describes (the abstract syntax of) a type system with a trivial type $\mathbb{1}$, a non-further specified collection of base types $b$, and four type constructors $(\cdot)$ for record types, $\{\cdot\}$ for finite set types, $[t]$ for list types, and $\oplus$ for union types. Record and union types use field labels $a_i$.

The semantics of such types is basically described by their domain, i. e. sets of values $dom(t)$. Usually, for a base type $b$ such as *Cardinal*, *Decimal*, *Float*, etc. the domain is some commonly known at most countable set with a common presentation. The domain of the trivial type contains a single special value, say $dom(\mathbb{1}) = \{\bot\}$. For constructed types we obtain the domain in the usual way:

$$dom((a_1 : t_1, \ldots, a_n : t_n)) = \{(a_1 : v_1, \ldots, a_n : v_n) \mid a_i \in dom(t_i) \text{ for } i = 1, \ldots, n\}$$
$$dom(\{t\}) = \{A \mid A \subseteq dom(t) \text{ finite}\}$$
$$dom([t]) = \{[v_1, \ldots, v_k] \mid v_i \in dom(t) \text{ for } i = 1, \ldots, k\}$$
$$dom((a_1 : t_1) \oplus \cdots \oplus (a_n : t_n)) = \bigcup_{i=1}^{n} \{(a_i : v_i) \mid v_i \in dom(t_i)\}$$

In particular, a union type $(a_1 : \mathbb{1}) \oplus \cdots \oplus (a_n : \mathbb{1})$ has the domain $\{(a_1 : \bot), \ldots, (a_n : \bot)\}$, which can be identified with the set $\{a_1, \ldots, a_n\}$, i. e. such types are in fact enumeration types.

In addition to the types, the functional description of a service operation includes pre- and post-conditions, which are defined by (first-order) predicate formulae. These formulae may contain further functions and predicates, which are subject to further (categorical) description. For instance, the terminology may comprise the following axioms:

$$\begin{aligned}
&\text{Service_Operation} \sqsubseteq \forall \text{pre.Condition} \sqcap \leq 1.\text{pre} \\
&\qquad \sqcap\ \exists \text{post.Condition} \sqcap \leq 1.\text{post} \\
&\text{Condition} \sqsubseteq \text{Formula} \sqcap \\
&\qquad \forall \text{uses.(Predicate} \sqcap \text{Function)} \\
&\text{Predicate} \sqsubseteq \exists \text{in.Type} \sqcap \leq 1.\text{in} \sqcap \neg \geq 1.\text{out} \\
&\text{Function} \sqsubseteq \exists \text{in.Type} \sqcap \leq 1.\text{in} \sqcap \\
&\qquad \exists \text{out.Type} \sqcap \leq 1.\text{out}
\end{aligned}$$

There are no general requirements for the categorical description, as it depends completely on the application domain. However, it will always lead to subconcepts of the concept Service_Operation plus additional concepts and roles. It will also add more details to the predicates and functions used in the pre- and post-conditions.

**Service-Level Agreements**
SLAs have been widely discussed in the literature. By now around 20 different types of SLAs have been identified by Rady (2012), and depending on the viewpoint these SLAs have been differently classified. Following our discussion in the introduction we consider that the main purpose of SLAs is to determine as precisely as possible the rights and obligations that govern the relationship between a service providers, service users, and, if applicable, third parties. Technically, our approach consists of three parts:

- an extension of the service ontology describing the content of the SLAs,
- a contracting framework that permits a contract skeleton to be extracted from the ontology, and
- a monitoring system that can be used to check, when a violation to an SLA has occurred.

As the service ontology is realised by some description logic, the contracting framework can be realised by queries against the ontology. This has been discussed by Rady (2014) using SPARQL to extract fragments of contracts from an SLA ontology. We dispense with discussing this aspect any further in this chapter. Also, as stated in

the introduction, SLA monitoring is still in an infant state, so we will not discuss it here, but we emphasise again that SLAs that cannot be monitored are de facto useless, i. e.*monitoribility* of SLAs is a necessary property of services.

Different from the classification by Rady we use the following classification schema, which puts a stronger emphasis on rights and obligations:

**Terms of Usage:** Some SLAs simply define general facts about the usage of services. Among these are the pricing schema, conditions for termination and suspension, and the applicable jurisdiction. In particular, these facts do not require to be monitored. They will appear as part of the contract extracted from the ontology and can be used to check bills or determine legal actions in case of inaccuracies.

**Technical Aspects:** Some SLAs refer to technical properties of the services that are determined by the service model, i. e. they are not SLAs in the proper sense. These technical aspects cover two different areas:

***Implementation Aspects:*** For instance, *portability* refers to the property of a service to be moved from one environment to another one, which is a property of the implementation. The same applies to *interoperability*, i. e. the property that the service can be combined with others, *scalability*, i. e. that the service can be applied to various input sizes, and *modifiability*, i. e. the property that the user may tune the service.

Furthermore, properties such as *testability*, *maintainability* and *verifiability* refer to software-technical characteristics. Actually, for a service user it is much more important that a service has been adequately tested and verified, the results of these quality assurance measures are available and reproducible, and preferably the service has already been certified according to some common quality standard rather than obtaining knowledge that verification, validation and testing can be done.

***Usage Aspects:*** This is usually associated with a usability SLA. Terms such as understandability or learnability used in this context are only vaguely defined and thus cannot be used for monitoring purposes. Usability studies can nonetheless give recommendations to service users.

However, some of the technical aspects, in particular the implementation aspects, may also be regarded as defining obligations and commitments of the provider to guarantee particular features of the service, in which case the scope of the commitments made has to become part of the SLA.

**Obligations and Rights of the Provider:** The most relevant class of SLAs covers obligations of the service provider, which also capture what the provider is committed to provide. The most commonly discussed SLAs in this class comprise the following:

***Availability:*** The SLA should cover when and for how long the service is guaranteed to be available to the user. Normally, this is formulated by some form of acceptable down-time. We will discuss availability SLAs further down.

***Performance:*** The SLA defines the expected (maximum / average) response time and throughput. Same as for availability, probability distributions could be used, but this is not state-of-the-art.

***Security and Privacy:*** SLAs concerning security and privacy could define the used methods for authentication, identity management, firewall rules, secrets to be preserved, confidentiality regulations, auditing procedures, etc. In our point of view it appears advisable not only to register the obligations of the provider but also the means to be taken to ensure security and privacy.

***Reliability:*** This refers to the measures taken by the provider to ensure that message content and the service results as a whole are reliable.

***Penalty and Compensation:*** These SLA capture mainly factual data about the amount to be repaid in case an SLA cannot be satisfied.

In addition they may be SLAs capturing rights of the provider, e.g. to close down the service for maintenance or in case of imminent security threats – this could be coupled with alerting obligations – to update the service to a new version or release, to cancel a contract in parts or as a whole, to increase prices, etc.

**Obligations and Rights of the User:** Analogously, SLAs may refer to obligations of the users in particular with respect to usage and security / privacy regulations. These may also be subject to penalty and compensation regulations.

### 3.3 A Perspective for Ambient Assistance

Buga et al. (2017b) discuss requirements for a robotic care system based on the commercially available GrowMu robot(see http://www.growmeup.eu/index.php/home/growmu-robot) with various extensions. The robot is to support elderly people in everyday life activities. While the core of the care system, the robot, acts autonomously and cooperates with its client, a key feature is its connection to a cloud to process information, increase its knowledge, and to share information with other care robots.

On a structural level the care robot is able to move around in the elderly person's household. It provides means for audio-visual interaction comprising a tablet screen for additional input/output of information, a camera to monitor the household and the client in order to detect critical situations (e. g. special requests of the client, reminders or alerts, etc.), a microphone for receiving requests by the client and enabling the discovery of critical situations by analysing sound signals (e. g. a cry, the sound of a falling object, etc.), and speakers to deliver sound messages. Furthermore, the robot is equipped with sensors for temperature and humidity, and with a tray attached to its body for delivering small items (e. g. drugs the client has to take).

Abstracting from the physical machine level comprising among others the electric drive, the sensors and the input/output devices, we consider the core of the care system as a service-oriented system. Thus, on the structural level we can think of services such as Bring($\vec{x}$), Come, Report($t(\vec{x})$), Alert($\vec{x}$), Observe($t(\vec{x})$) and Tacit!, respectively. When issued the robot is to bring the small item $\vec{x}$ to the client, move to the client and wait for further instructions, record the kind of task $t(\vec{x})$, the time of issuing, the issuer, and (if applicable) any rationale for the task, deliver audio-visual information to the client and send the information to the cloud, record information about an activity $t$ (with parameters $\vec{x}$), or stop the transmission of observations to the cloud, respectively (for details see Buga et al. 2017b). All these structural services assume a proper functioning of the robot.

On an operational level the care robot is to learn the elderly person's needs and habits over time and enhance its functionality, which permits compensation for the degradation of the client's capabilities, and supports encouragement for remaining active, independent and socially involved. For this the cloud maintains an anonymised profile of the client comprising routine activities, special care needs, risks that require observation, particular interests, etc. The observations collected by the robot are subject to machine learning mechanisms that enable to learn changes to the profile. All services provided by the care robot depend on such profiles. Profiles are to be maintained by service knowledge bases in the cloud. All robot-cloud interaction will exploit a client-cloud middleware. The middleware is used to support the interaction of a robot with cloud-based services that are used in several ways such as failure alerts, failure compensation, anticipation of failure situations and failure prevention, behavioural pattern detection, safety hazards detection, and enabling of basic security mechanisms.

In order to support optimal care the cloud-based robotic care system is to support multiple interacting care robots. This addresses a learning aspect as well as a collaboration aspect. These are used in the various ways dealing with uncertainty

and privacy and enabling collaboration among robots. For instance, in the case that multiple robots take care of a group of elderly people, e.g. in an elderly home, each robot having dedicated tasks (e.g. being company during a walk, delivering post or medicine, remind a caretaker to drink, etc.), the robots have to interact with each other and decide (by means of consensus algorithms), which robot is responsible for a particular patient. These responsibilities may change over time, for which location-based information will be required from the robot, that is analysed by the monitoring layer to anticipate the upcoming needs of the patient, and the adaptation layer will realise the change of responsibilities when needed. This is extended further to capture failure situations, where other robots will have to step in to replace the functionality of a broken down "colleague".

## 4 A Proposal for Continued Research

We reported on our research on the foundations of conceptual modelling for service-oriented systems. The BDCM$^2$ framework (reported in detail by Schewe and Wang 2015) addresses in an axiomatic way the following important features of services:

***Behaviour*:** There must exist a general behavioural theory of services. For this we outlined our research on the model of Abstract State Services (AS$^2$s) by Ma et al. (2009a), which follows the line of research of the ASM thesis.

***Description*:** There must exist a description of a service that allows it to be discovered and used. For this we stressed services ontologies, e. g. the model by Ma et al. (2009b) addressing functional, categorical and quality aspects of services.

***Contracting*:** There must exist a contract between service provider and user covering all relevant SLAs for the service. Here we outlined that quality aspects of services should be extended to capture also all other aspects that could give rise to SLAs. The collection of SLAs has to be treated as a binding contract between service provider and user.

***Mediation*:** A user must use services in the contracted way, but can build service mediations to realise service-oriented systems satisfying his purposes. For this we rely on the model of service mediators developed by Ma et al. (2012).

***Monitoring and Adaptation*:** It must be possible to monitor the execution of service mediator instance in order to validate its behaviour and contracted SLAs. In case of detected critical situations or SLA violations the system configuration should be changed on-the-fly.

We then sketched the modelling of distributed, adaptive systems that are based on services supported by multiple clouds (for details see Buga et al. 2017a and the references in there). The underlying model for service-oriented systems that exploit cloud-enabled services is the mediator model, and the selection of such services is driven by a service ontology comprising functional, categorical and SLA-based characteristics. The concrete interaction with the multiple clouds is realised by a middleware architecture developed by Bósa et al. (2014). This middleware is extended by monitoring and adaptation layers that identify the need for a change of a mediator instantiation and provide an updated one. All parts of the model have been specified using Abstract State Machines including the extensions covering ambient computing (see Börger et al. 2012), concurrency (see Börger and Schewe 2016) and linguistic reflection (see Schewe et al. 2017). We further illustrated the model by a use case concerning a cloud-based robotic care system providing services for the support of an elderly client (see Buga et al. 2017b for more details).

What is more exciting for an active researcher than an open invitation to contribute to still open problems? The BDCM$^2$ framework tries to cover all aspects of software services and goes much further than related work by Bergholtz et al. (2015), Ferrario et al. (2011), Sampson and Froehle (2006), Preist (2004) or Alves et al. (2007). Nonetheless, there is still no common agreement on the ontological question what services and

service-oriented systems *are*. For instance, the proposal by Dahanayake and Thalheim (2015) contains ideas that are worth being considered as well. It would be great, if aspects that have not yet been covered adequately in the BDCM$^2$ framework (if any) were discovered and the framework fine-tuned. Let us make cloud-enabled distributed adaptive systems a prime theme for conceptual modelling and use ambient assistance as an application, where service-orientation, autonomous systems and domain-specific conceptual modelling come together.

## References

Al Machot F., Mayr H. C., Michael J. (2014) Behavior Modeling and Reasoning for Ambient Support: HCM-L Modeler. In: Ali M., Pan J., Chen S., Horng M. (eds.) Modern Advances in Applied Intelligence (IEA/AIE 2014). Lecture Notes in Computer Science Vol. 8482. Springer, pp. 388–397

Alonso G. et al. (eds.) Web Services: Concepts, Architectures and Applications. Springer-Verlag

Alves A. et al. (2007) Web Services Business Process Execution Language, version 2.0. Last Access: OASIS Standard Committee, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

Arsanjani A., Ghosh S., Allam A., Abdollah T., Ganapathy S., Holley K. (2008) SOMA: A method for developing service-oriented solutions. In: IBM Systems Journal 47(3), pp. 377–396

Benatallah B., Casati F., Toumani F. (2006) Representing, Analysing and Managing Web Service Protocols. In: Data and Knowledge Engineering 58(3), pp. 327–357

Bergholtz M., Andersson B., Johannesson P. (2015) Towards a model of services based on co-creation, abstraction and rights distribution. In: Thalheim B., Schewe K.-D., Prinz A., Buchberger B. (eds.) Correct Software in Web Applications and Web Services. Springer, pp. 29–44

Börger E., Schewe K.-D. (2016) Concurrent Abstract State Machines. In: Acta Informatica 53(5), pp. 469–492

Börger E., Stärk R. (2003) Abstract State Machines. Springer-Verlag, Berlin Heidelberg New York

Börger E., Cisternino A., Gervasi V. (2012) Ambient Abstract State Machines with Applications. In: J. Comp. Syst. Sci. 78(3), pp. 939–959

Bósa K. (2012) Formal Modeling of Mobile Computing Systems Based on Ambient Abstract State Machines. In: Semantics in Data and Knowledge Bases. LNCS Vol. 7693. Springer, pp. 18–49

Bósa K. (2013) An Ambient ASM Model for Client-to-Client Interaction via Cloud Computing. In: Proceedings of the 8th International Conference on Software and Data Technologies (ICSOFT). SciTePress, pp. 459–470

Bósa K., Chelemen R., Vleju M. B. (2015) A Formal Model of Client-Cloud Interaction. In: Thalheim B., Schewe K.-D., Prinz A., Buchberger B. (eds.) Correct Software in Web Applications. Springer, pp. 83–144

Bósa K., Holom R. M., Vleju M. B. (2014) A Formal Model of Client-Cloud Interaction. In: Thalheim B., Schewe K.-D., Prinz A., Buchberger B. (eds.) Correct Software in Web Applications and Web Services. Springer, pp. 83–144

Brodie M. L. (1984) On conceptual modelling – perspectives from artificial intelligence, databases and programming languages. Topics in information systems. Springer

Buga A., Nemeş S. T., Schewe K.-D. (2017a) Conceptual Modelling of Autonomous Multi-cloud Interaction with Reflective Semantics. In: Mayr H. C., Guizzardi G., Ma H., Pastor O. (eds.) Conceptual Modeling - 36th International Conference (ER 2017). Lecture Notes in Computer Science Vol. 10650. Springer, pp. 120–133

Buga A., Nemeş S. T., Schewe K.-D. (2017b) Towards Care Systems Using Model-Driven Adaptation and Monitoring of Autonomous Multi-clouds. In: de Cesare S., Frank U. (eds.) Advances in Conceptual Modeling (ER 2017 Workshops). Lecture Notes in Computer Science Vol. 10651. Springer, pp. 26–35

Dahanayake A., Thalheim B. (2015) W*H: The Conceptual Model of Services. In: Thalheim B., Schewe K.-D., Prinz A., Buchberger B. (eds.) Correct Software in Web Applications. Springer, pp. 145–176

Erl T. (2007) SOA: Principles of Service Design. Prentice Hall Press, Upper Saddle River, NJ, USA

Fensel D. et al. (2007) Enabling Semantic Web Services. Springer-Verlag

Ferrario R., Guarino N., Fernández-Barrera M. (2011) Towards an Ontological Foundation for Services Science: The Legal Perspective. In: Sartor G., Casanovas P., Biasiotti M., Fernández-Barrera M. (eds.) Approaches to Legal Ontologies. Law, Governance and Technology Vol. 1. Springer, Netherlands, pp. 235–258

Ferrarotti F., Schewe K.-D., Tec L., Wang Q. (2016) A New Thesis Concerning Synchronised Parallel Computing – Simplified Parallel ASM Thesis. In: Theoretical Computer Science 649, pp. 25–53

Fliedl G., Kop C., Mayr H. C. (2005) From textual scenarios to a conceptual schema. In: Data Knowl. Eng. 55(1), pp. 20–37

Grau B. C., Horrocks I., Motik B., Parsia B., Patel-Schneider P. F., Sattler U. (2008) OWL 2: The next step for OWL. In: Journal of Web Semantics 6(4), pp. 309–322

Hesse W. et al. (2004) Ontologien in der und für die Softwaretechnik. In: Rumpe B., Hesse W. (eds.) Modellierung 2004. LNI Vol. 45. GI, pp. 269–270

Hull R., King R. (1987) Semantic Database Modeling: Survey, Applications, and Research Issues. In: ACM Comput. Surv. 19(3), pp. 201–260

Kaschek R. et al. (2006) Information systems design: through adaptivity to ubiquity. In: Inf. Syst. E-Business Management 4(2), pp. 137–158

Kossak F. et al. (2016) Hagenberg Business Process Modelling Method. Springer

Lampesberger H., Rady M. (2015) Monitoring of Client-Cloud Interaction. In: Thalheim B., Schewe K.-D., Prinz A., Buchberger B. (eds.) Correct Software in Web Applications. Springer, pp. 177–228

Lampesberger H. (2016) Technologies for Web and cloud service interaction: a survey. In: Service Oriented Computing and Applications 10(2), pp. 71–110

Ma H., Schewe K.-D., Thalheim B., Wang Q. (2008) Abstract State Services. In: Song I.-Y. et al. (eds.) Advances in Conceptual Modeling – Challenges and Opportunities, ER 2008 Workshops. LNCS Vol. 5232. Springer-Verlag, pp. 406–415

Ma H., Schewe K.-D., Thalheim B., Wang Q. (2009a) A Theory of Data-Intensive Software Services. In: Service Oriented Computing and Its Applications 3(4), pp. 263–283

Ma H., Schewe K.-D., Thalheim B., Wang Q. (2012) A Formal Model for the Interoperability of Service Clouds. In: Service Oriented Computing and Its Applications 6(3), pp. 189–205

Ma H., Schewe K.-D., Wang Q. (2009b) An Abstract Model for Service Provision, Search and Composition. In: Kirchberg M. et al. (eds.) Services Computing Conference - APSCC 2009. IEEE Asia Pacific, pp. 95–102

Mayr H. C. et al. (2007) Business Process Modeling and Requirements Modeling. In: First International Conference on the Digital Society (ICDS 2007), p. 8

Michael J., Mayr H. C. (2013) Conceptual Modeling for Ambient Assistance. In: Ng W., Storey V. C., Trujillo J. (eds.) Conceptual Modeling - 32th International Conference (ER 2013). Lecture Notes in Computer Science Vol. 8217. Springer, pp. 403–413

Michael J., Mayr H. C. (2016) The Process of Creating a Domain Specific Modelling Method (Extended Abstract). In: Mendling J., Rinderle-Ma S. (eds.) Proceedings of the 7th International Workshop on Enterprise Modeling and Information Systems Architectures (EMISA 2016). CEUR

Workshop Proceedings Vol. 1701. CEUR-WS.org, pp. 40–43

Universal Description, Discovery and Integration (UDDI). Last Access: http://www.uddi.org

Papazoglou M. P., van den Heuvel W.-J. (2006) Service-oriented design and development methodology. In: International Journal of Web Engineering and Technology 2(4), pp. 4120–442

Papazoglou M. P., van den Heuvel W.-J. (2007) Service Oriented Architectures: Approaches, Technologies and Research Issues. In: VLDB Journal 16(3), pp. 389–415

Preist C. (2004) A Conceptual Architecture for Semantic Web Services. In: McIlraith S. A., Plexousakis D., van Harmelen F. (eds.) The Semantic Web – ISWC 2004. Lecture Notes in Computer Science Vol. 3298. Springer, Berlin Heidelberg, pp. 395–409

Rady M. (2012) Parameters for Service Level Agreements Generation in Cloud Computing: A Client-Centric Vision. In: Castano S. et al. (eds.) Advances in Conceptual Modeling – ER 2012 Workshops. Lecture Notes in Computer Science Vol. 7518. Springer, Berlin Heidelberg, pp. 13–22

Rady M. (2014) Generating An Excerpt Of A Service Level Agreement From A Formal Definition of Non-Functional Aspects using OWL. In: Journal of Universal Computer Science 20(3), pp. 366–384 https://doi.org/10.3217/jucs-020-03-0366

Sampson S. E., Froehle C. M. (2006) Foundations and Implications of a Proposed Unified Services Theory. In: Production and Operations Management 15(2), pp. 329–343

Schewe K.-D. et al. (2005) A Conceptual View of Web-Based E-Learning Systems. In: EAIT 10(1-2), pp. 83–110

Schewe K.-D., Ferrarotti F., Tec L., Wang Q., An W. (2017) Evolving Concurrent Systems – Behavioural Theory and Logic. In: Proceedings of the Australasian Computer Science Week (ACSW 2017). ACM, Deakin University, Victoria, Australia, 77:1-77-10

Schewe K.-D., Thalheim B. (2018) Design and Development of Web Information Systems. (to appear). Springer

Schewe K.-D., Wang Q. (2010) A Customised ASM Thesis for Database Transformations. In: Acta Cybernetica 19(4), pp. 765–805

Schewe K.-D., Wang Q. (2012) Preferential Refinements of Abstract State Machines for Service Mediators. In: Muccini H., Tang A. (eds.) Proc. QSIC 2012. IEEE CPS, pp. 158–166

Schewe K.-D., Wang Q. (2015) What Constitutes a Service on the Web? In: Thalheim B., Schewe K.-D., Prinz A., Buchberger B. (eds.) Correct Software in Web Applications and Web Services. Springer, pp. 257–292

Shekhovtsov V. A., Mayr H. C., Kop C. (2012) Towards Conceptualizing Quality-Related Stakeholder Interactions in Software Development. In: Mayr H. C., Kop C., Liddle S. W., Ginige A. (eds.) Information Systems: Methods, Models, and Applications (UNISCON 2012). Lecture Notes in Business Information Processing Vol. 137. Springer, pp. 73–86

Simple Object Access Protocol (SOAP). Last Access: http://www.w3c.org/TR/soap

Thalheim B. (2000) Entity-Relationship Modeling: Foundations of Database Technology. Springer-Verlag

Thalheim B., Schewe K.-D., Prinz A., Buchberger B. (eds.) Correct Software in Web Applications and Web Services. Springer

Zeithaml V. A., Parasuraman A., Berry L. L. (1985) Problems and Strategies in Services Marketing. In: Journal of Marketing 49(2), pp. 33–46

# Evaluating User Behavior as They Create Mappings in a Web Development System Using Local Radiance

Scott Britell[*,a], Lois M.L. Delcambre[a]

[a] Computer Science Department, Portland State University, Portland, OR, USA

Abstract. *Information integration with local radiance (IILR) is a system designed for use in web development frameworks that allows for the creation of polymorphic widgets based on small schema fragments and mappings to local schema that allow non-expert users to instantiate these widgets in their site. Here, we present results of a user study using IILR. We show that non-expert users can, and for the most part enjoy, creating the mappings required for our system. We describe the different behaviors observed of our participants and relate these behaviors to the survey data from our users.*

Keywords. Conceptual Modeling • Web Modeling • User Study

## 1 Introduction

The goal of our work is to facilitate and empower data creators and domain experts to perform complex tasks in web-based information systems that heretofore needed a database or web developer to accomplish. Modern web-based content management systems have enabled non-technical, domain-savvy users to store and publish data in rich structures with content types that can reference other content types. This effectively allows non-technical, domain users to define and populate their own conceptual models. While most domain users can publish their data in complex structures, configuring sites to use widgets typically requires expert developers. These widgets are often limited in their flexibility: the user's data must either fit the existing schema of the widget or the widget must be rewritten to accommodate the user's schema. Our approach provides more complex widgets that are written generically against widget-specific schemas (*canonical structures*). These widgets can then be configured by non-technical, domain experts by creating simple mappings between their data in the schemas that they have created (*local schemas*) and domain-specific schemas (*domain structures*) that are isomorphic to the canonical structures.

Our previous work (Britell et al. 2014, 2016) presented the formal basis for our system and its implementations. Our IILR technology with generic widgets has been implemented in an operational, production website[1] with 6,500+ documents and 4,000+ users for over six years but without the ability for domain users to provide the mappings. In this work we present the results of our first user study where domain users with a range of technical expertise are asked to provide mappings. Subjects were provided a short training session and then required to use our system to create mappings for a widget in a website that they had not seen before. We show that all subjects of the study were able to successfully use the system. Subjects used the system in a number of different ways and we report on these different behaviors. The overriding goal of the study was to evaluate the feasibility of

* Corresponding author.
E-mail. britell@cs.pdx.edu
This work was supported in part by National Science Foundation grants 0840668 and 1250340. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

[1] http://stemrobotics.cs.pdx.edu

our approach; we show that users generally were confident of their choice of mappings and enjoyed using our system.

This paper provides a brief overview of the IILR system in Section 2. Section 3 describes the design of our user study, including the participants, the test structure, the websites used, and the mapping interface used. We explore the different ways users interact with our system in Section 4. Results of the user surveys and their relationship to user behaviors are shown in Section 5. The paper concludes with a short discussion in Section 6.

## 2 Background and Related Work

Traditional structured information integration scenarios require that local schemas be mapped to a global schema; developers (and users) can then issue queries against the global schema to retrieve information from all participating local data sources. In our work, we share the goal of allowing multiple local schemas because we have seen that even within a single website, different groups of users may define distinct content types/relationships to describe semantically similar content. Thus our work supports information integration from multiple user-defined schemas (what we call local schemas). Note that one additional feature of our work is that we are able to extract the local schema names (based on the mappings in place) for display in our widgets - rather than providing query answers (against the global schema) that use only the global schema names. This feature that allows the local schema names to "shine through" to the global or integrated level is why we call our work Information Integration with Local Radiance.

Our work differs from traditional information integration approaches in that we use three levels of schemas, as shown in Figure 1. A generic schema (canonical structure) used by the widget developer, a domain-specific schema (domain structure) that is presented to the user defining the mappings, and the local schemas as defined by the domain users. We describe each of these in turn. We use a simple conceptual model where content types/entities are shown as labeled rectangles and uni- and bidirectional references from one content type/entity type to another are shown as uni- and bidirectional arrows, respectively.

### 2.1 Canonical/Domain Structures and Local Schemas

An example of a canonical structure is shown at the bottom of Figure 1 to support a navigation widget, a preview of which is shown in Figure 2. The canonical structure represents only the essential structure required to write the widget code. For this example, the navigation widget presents the initial entity (i.e., Parent) with a nested display of all subordinate entities (i.e., Part). The navigation widget recursively displays further subordinate entities using the Parent - Part relationship, for all mappings that have been provided. Note that the canonical structure uses names related to the functionality of the widget and not to the domain of the application (library collections, in this example). The widget is thus domain-independent and can be reused in other domains.

A domain structure for use in this website is shown in the middle of Figure 1. This domain structure is isomorphic to the canonical structure shown at the bottom of the figure. A domain structure provides domain-meaningful names for the schema elements in the corresponding canonical structure.

A local schema for a library is shown at the top of Figure 1. Here we see that each Library references any number of Collections containing (i.e., referencing) any number of Books. Each Book may contain/reference Chapters which then may contain/reference Sections.

### 2.2 Mappings

Our system requires mappings at two levels as shown in Figure 1. One level of mappings is between the canonical and domain structures—shown as dotted black lines between the domain structure in the middle and the canonical structure at the bottom of the figure. We envision that the widget developer or a person charged with configuring widgets for a particular application

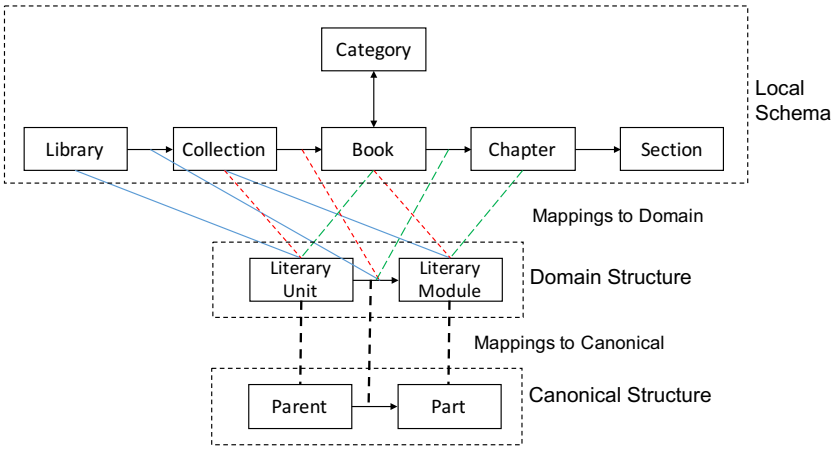



*Figure 1: A local library schema (top), the library domain structure (middle), the parent-part canonical structure (bottom), and mappings between the three.*

domain would define domain structures and also provide the mappings from the canonical structure used by a widget and the corresponding domain structure.

The focus of this paper is on the mappings from each local schema to the relevant domain structure (that has already been mapped to the canonical structure for the widget). An example of how the local schema for the library (at the top of the figure) could be mapped to the domain structure (in the middle of the figure) is shown with colored lines. The blue solid lines show that the Library-Collection portion of the local schema has been mapped to the Literary Unit-Literary Module portion of the domain structure. Similarly, the red dotted lines show that the Collection-Book portion of the local schema has also been mapped to the Literary Unit-Literary Module portion of the domain structure. And, the green dotted lines show the Book-Chapter portion of the local schema mapped to the Literary Unit-Literary Module portion of the domain structure. The navigation widget accordingly displays a library (The Metropolitan Library) with two collections (Astrology and Family Studies) where the Calendars books in the Astrology Collection has two chapters shown, see Figure 2. Note that the navigation widget displays the +/− symbol that allows the end-user to expand/contract the navigational display, as desired.

## 2.3 Widgets

To summarize, the navigational widget, as shown in Figure 2 is written to query the canonical structure but is able to display the local data and schema because of the two levels of mapping provided, as shown in Figure 1. The widget always displays the data as mapped; if a mapping is added or removed, the widget immediately displays more or less information to the user.

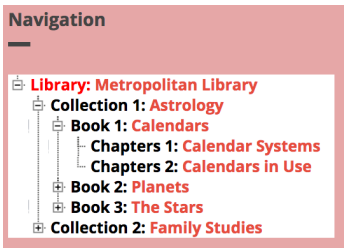



*Figure 2: Widget using mappings from Figure 1*

## 2.4 Related Work

Our work has been strongly influenced by work in schema mapping (Dessloch et al. 2008) and ETL (Miller et al. 2001) where users can draw simple lines between source and target schemas. While these tools are automated to facilitate the

schema mapping process, the mappings themselves and the tools to use them are targeted at expert database developers. We adopt this approach to allow non-expert users to do similar simple mappings.

We are also inspired by the field of end-user programming (Jones 1995; Lieberman et al. 2006) whose goal is to get non-developer users of software to create, modify, and extend that software. Current end-user web programming paradigms (Rode et al. 2005; Wong and Hong 2007) focus on allowing end-users to create "mashups" of existing widgets and data on the internet. We focus instead on letting end-users populate polymorphic widgets with their own data and schema.

We also believe that we can exploit the domain knowledge of our users to facilitate this process much in the same way it can be used to get non-expert users to perform semantic annotation (Hinze et al. 2012; Price et al. 2009). Our work is similar to approaches in conceptual models that introduce intermediate models between end-user specifications and system models (Mayr and Kop 2002; Vöhringer and Mayr 2006).

## 3 Design of the User Study

We designed our user study to test if subjects could create mappings between local schemas and domain structures. Subjects were asked to complete three tasks; one training task in which participants were guided through the process of creating mappings in a site with a fairly simple schema; and, two tasks where participants worked on their own to create mappings in two different sites (with a simple schema and a more complex schema). Subjects were given a demographic questionnaire at the beginning of the session, evaluation questionnaires at the end of each of the two testing tasks, and an overall evaluation questionnaire at the end of the session. As we expect our tool to be used by domain savvy users our test was limited to sites in a single domain (in this case, an educational domain).

For the training task, subjects used a website built using the library local schema shown in Figure 1. This schema has a simple hierarchy between Library, Collection, Book, Chapter, and Section. There is also a bidirectional relationship between Category and Book so that the subjects could create recursive mappings using the tool. The subjects were shown the structure of the site using only the hypertext links in the webpages within the operational website.

They were then shown the mapping tool (Figure 3) that, for a given domain structure, allows the user to select a content type from a list of all possible content types in the site and then choose a relationship associated with that type. Part of the mapping interface is a preview widget that shows how the navigation widget in the site would look using the given mappings. The tool then allows users to delete a specific mapping, save all of their mappings, or delete all of their mappings (Figure 4).

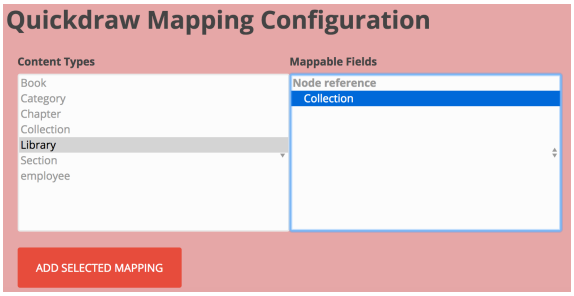



*Figure 3: In the mapping interface, users select a content type (on the left) and then are shown all possible relationships to other content types (on the right).*

In the training tasks the subjects were asked to create a number of specific mappings and encouraged to make additional mappings, as desired. There may be many different mappings that can be created within any given site for any number of reasons so we explicitly allowed our subjects to create whatever mappings they felt were appropriate. Since the choice of mappings is subjective, we did not test to see if subjects would create any specific mappings. Mappings were only deemed incorrect if the end result produced irregular widget behavior (e.g., duplicate mappings or disjoint

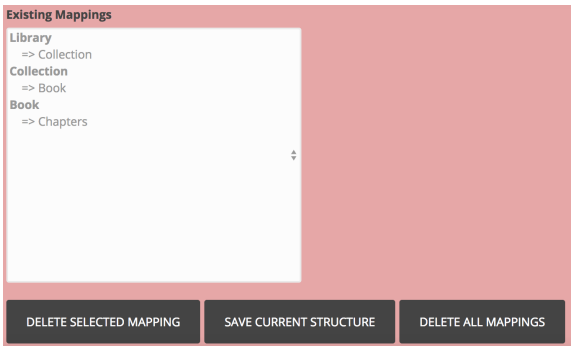



*Figure 4: The mapping tool allows users to save their current mappings, delete specific mappings, or delete all mappings.*

mappings). After the scripted part of the training session participants were allowed to explore the training site and the mapping tool for as long as they desired.

After the training task, participants were then asked to create mappings they saw as appropriate for a website with the schema shown in Figure 5. This schema is a simple hierarchy of an academic journal using unidirectional relationships only. For the second testing task participants were asked

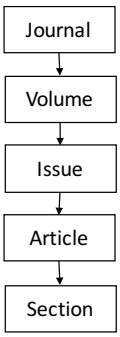



*Figure 5: Schema for first task in the study.*

to create mappings they saw as appropriate for a website based on a university schema shown in Figure 6. The schema was created with bidirectional relationships and cycles.

Participants for the study were recruited from the pool of departmental administration staff from the university who are in charge of updating the university webpage for their respective departments. All participants had working knowledge

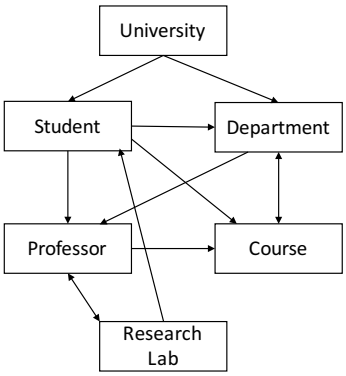



*Figure 6: Schema for second task in the study.*

of Journals, Libraries, and Universities. The participants had varying degrees of technical expertise ranging from three to more than ten years of website configuration experience and none to more than ten years of database experience.

## 4 User Behaviors

We showed participants how to browse the site, see a preview of the widget, and create large and small mappings. We emphasized the use of the preview functionality as we believed it would benefit the creation and checking of mappings.

Figure 7 shows an overview of the study subjects' sessions. Each subject's session is represented in a horizontal bar beginning with their anonymous id. The sessions are broken into boxes for each task, the first (pink) box shows the training task, the second (blue) box shows the first testing task, and the last (green) box shows the second testing task. The smaller boxes inside each box represent the various actions performed by the subject within the task (explained below). The longest session lasted a little less than 50 minutes and the shortest was less than 20 minutes. This is unsurprising given the open-ended nature of the tasks. In most sessions, subjects took a longer time with the second task likely due to the more complex nature of the local schema for the site in that task. For the two subjects who completed the second task, one created a single set of mappings for the university, without cycles, and decided they were done while the second appeared to rush

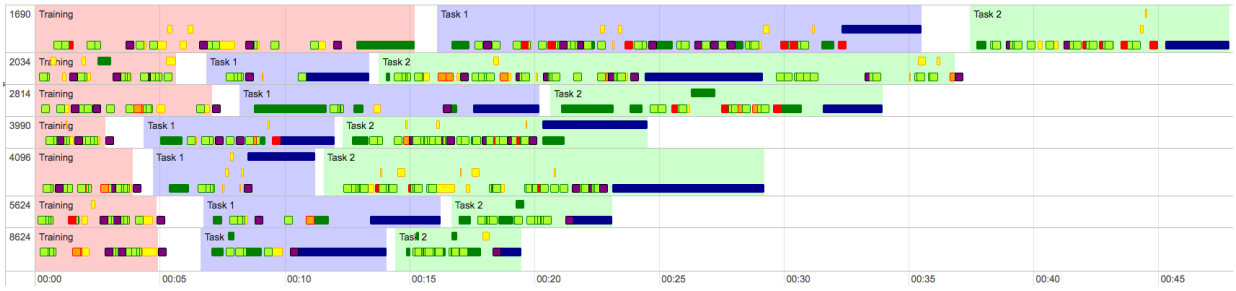



*Figure 7: Timelines of user sessions showing length of training, task 1, and task 2.*

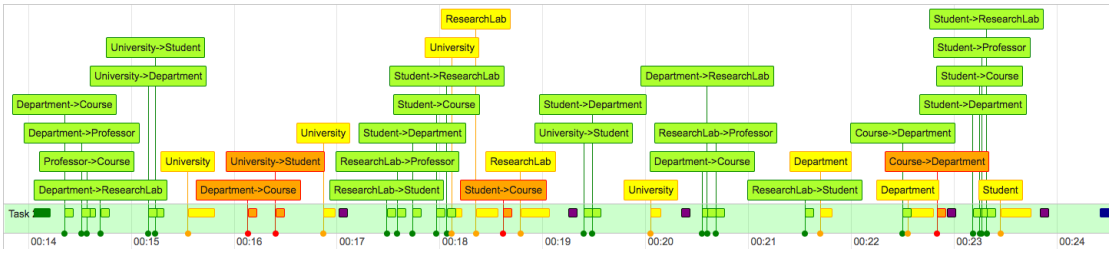



*Figure 8: A study session of task 2 demonstrating the preview, test and check, and the enity-centric behaviors.*

through the task and saved a set of mappings that included duplicate mappings. This was the only subject to save a set of mappings we deemed to be incorrect; other subjects created structures that had duplicate mappings or contained disjoint parts of the schema but in all cases these subjects discovered and deleted their bad mappings.

As mentioned above, we believed that the preview feature of our tool would be useful for checking and evaluating mappings. We found that some of our participants chose to use the preview feature while others chose to browse through the live site and see the live widget in the context of the actual webpages. Figure 8 shows an example of the use of the preview. In this case the subject starts the task by creating six mappings, uses the preview function to check the mappings, deletes two mappings, uses the preview again, and then saves the set of mappings. This subject continues creating a few mappings and checking with preview. Figure 9 shows another example of a previewer where the user starts by creating a single mapping, previews that mapping, and then saves the set of mappings. This user continues previewing after each new mapping. Contrast that with Figure 10 where the subject creates five mappings, browses the site, creates two more mappings, browses the site again, creates six more mappings, and then saves the set of mappings.

Figures 8 and 9 also demonstrate two other behavior patterns observed in the test. In Figure 8 the subject creates a few mappings, checks them with preview, then deletes a mapping or two before saving the set of mappings. Compare that with Figure 9 where the subject often creates a number of mappings and when they decide that they are incorrect or not to their liking they delete the entire set of mappings and start over.

The three examples shown thus far also demonstrate the two different ways subjects approached the process of mapping. In Figures 8 and 10 the subjects created larger navigational structures (with more mappings). Theses structures were also built in an entity-centric way, where the subject starts at one type, creates all the mappings related to that type and then moves to the next. Contrast that to Figure 9 where the subject creates many small structures and the mappings are often created in what appears to be a random fashion.

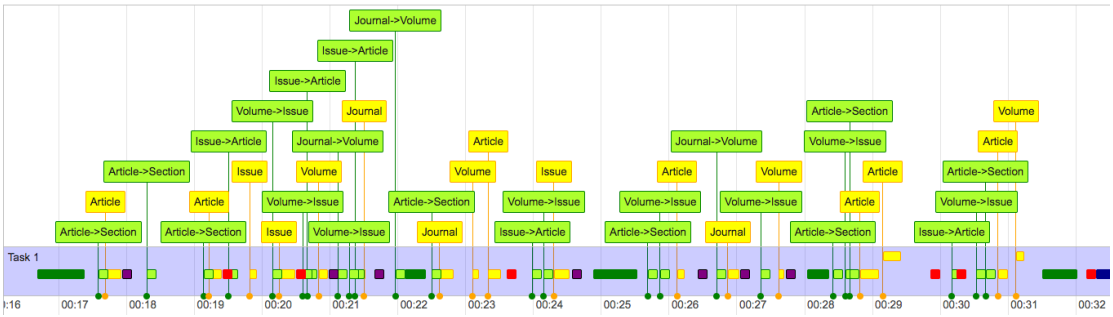



*Figure 9: A study session of task 1 demonstrating the preview, delete and start over, and the random behaviors.*

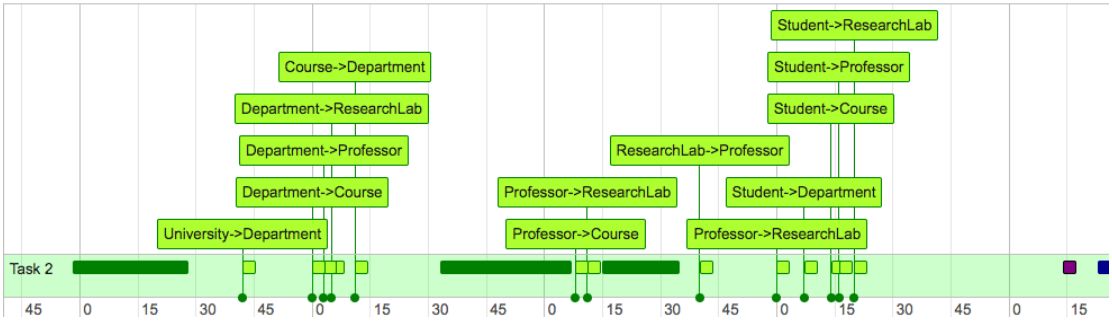



*Figure 10: A study session of task 2 demonstrating the browse, large, and entity-centric behaviors. Darker rectangles represent browsing behavior, mappings are displayed above the lighter colored squares, and the dark square at the end saves all mappings.*

## 5 Results

In regards to our main goal of this study—to see if domain savvy users can perform the mapping tasks in our system—the study was a success. All participants were able to complete the given tasks within a reasonable timeframe (we designed the test to take no more than one hour with the caveat that the open-ended nature could have made it take longer) and, although they could leave at any time during the test, no one left the test prematurely. (One participant inadvertently failed to complete the final questionnaire but completed all tasks and task questionnaires.) Participants were asked to rate the overall usefulness of the tool and enjoyment of mapping on a scale of one (Strongly Disagree) to five (Strongly Agree) with the average response for both questions being 3.7.

An interesting aspect of the behaviors listed above is how groups of differing behaviors corresponded to satisfaction results of the tool and mapping. Table 1 shows some of this detail. We found that the group of subjects that used preview was also the group with more than five years of experience. Those with less experience tended to exhibit more browsing behavior.

*Table 1: Aggregated user feedback showing satisfaction with the system for each task and overall and a scale of one (Strongly Disagree) to five (Strongly Agree).*

| | Satisfaction | | |
|---|---|---|---|
| Behavior Group | Task1 | Task2 | Overall |
| Previewer and Experienced | 2.8 | 3.4 | 3.6 |
| Non Previewer and Inexperienced | 5 | 2 | 4 |
| Larger and Entity-centric | 4 | 3.25 | 4.7 |
| Smaller and Random | 2.7 | 2.7 | 2.7 |

Overall those with less technical experience tended to find the system more useful than those with more experience. In the questionnaires related to the tasks, those participants with more experience expressed some frustration that they were limited to what the tool could do when they knew how to edit HTML directly to get the results they wanted. Also of note is that even though inexperienced users preferred the tool overall they were less satisfied during the second more complex task.

Table 1 also shows the difference between the larger/entity-centric mappers and the smaller/random mappers. We see that across the board the larger/entity-centric mappers were more satisfied with the tool on each of the tasks and overall. It is likely that these subjects had a better understanding of the structure of the sites, our tool, and the tasks.

## 6 Conclusions

We use three levels of schema to facilitate end-users as they enhance their websites by mapping their local schemas to domain-specific schemas while allowing developers to maintain their own widget-specific schemas. Our test successfully demonstrated that non-expert domain users can perform the mapping tasks necessary to use our system. Overall subjects enjoyed using the tool. It is interesting that those with less experience appear to like our tool more than those with more experience, but we had a very small sample size (seven) and the user interface for our mapping tool was rudimentary, at best.

Our production website that implements IILR is written using the Drupal[2] web development framework. We have made our technology, including the mapping tools, available publicly to the Drupal community. Given our experience with our production website over the years and the evolution of our thinking regarding the IILR approach, we look forward to re-implementing the system including new implementations in other web development frameworks. This would likely present a cleaner, more fully featured, and more general implementation of IILR.

[2] http://drupal.org

## References

Britell S., Delcambre L. M. L., Atzeni P. (2014) Flexible Information Integration with Local Dominance. In: Information Modelling and Knowledge Bases XXVI, pp. 21–40

Britell S., Delcambre L. M. L., Atzeni P. (2016) Facilitating Data-Metadata Transformation by Domain Specialists in a Web-Based Information System Using Simple Correspondences. In: Conceptual Modeling: 35th International Conference, ER 2016 Springer International Publishing, pp. 445–459

Dessloch S., Hernandez M. A., Wisnesky R., Radwan A., Zhou J. (2008) Orchid: Integrating Schema Mapping and ETL. In: ICDE '08: Proceedings of the 2008 IEEE 24th International Conference on Data Engineering. IEEE Computer Society, Washington, DC, USA, pp. 1307–1316

Hinze A., Heese R., Luczak-Rösch M., Paschke A. (2012) Semantic Enrichment by Non-Experts: Usability of Manual Annotation Tools. In: The Semantic Web – ISWC 2012: 11th International Semantic Web Conference. Springer International Publishing, pp. 165–181

Jones C. (1995) End-User Programming. In: Computer 28(9), pp. 68–70

Lieberman H., Paternò F., Klann M., Wulf V. (2006) End-User Development: An Emerging Paradigm. In: End User Development. Springer Netherlands, Dordrecht, pp. 1–8

Mayr H. C., Kop C. (2002) A User Centered Approach to Requirements Modeling. In: M. Glinz, G. Müller-Luschnat (Hrsg.): Modellierung 2002 - Modellierung in der Praxis - Modellierung für die Praxis Springer, pp. 75–86

Miller R. J., Hernández M. A., Haas L. M., Yan L., Howard Ho C. T., Fagin R., Popa L. (2001) The Clio project. In: ACM SIGMOD Record 30(1), pp. 78–83

Price S. L., Lykke Nielsen M., Delcambre L. M., Vedsted P., Steinhauer J. (2009) Using semantic components to search for domain-specific documents: An evaluation from the system perspective and the user perspective. In: Information Systems 34(8), pp. 724–752

Rode J., Bhardwaj Y., Pérez-Quiñones M. A., Rosson M. B., Howarth J. (2005) As Easy as "Click": End-User Web Engineering. In: Web Engineering: 5th International Conference, ICWE. Springer, Berlin, Heidelberg, pp. 478–488

Vöhringer J., Mayr H. C. (2006) Integration of Schemas on the Pre-Design Level Using the KCPM-Approach. In: Advances in Information Systems Development. Springer US, Boston, MA, pp. 623–634

Wong J., Hong J. I. (2007) Making mashups with marmite. In: Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '07. ACM Press, New York, New York, USA, p. 1435

# Eliciting User Interface Requirements and Deriving Usability Problems from Scenario Textual Descriptions

Josefina Guerrero-García[*,a], Juan González-Calleros[a]

[a] Computer Science Faculty, Benemérita Universidad Autónoma de Puebla, Puebla, Mexico.

Abstract. *Scenario Textual Descriptions (STD) are general-purpose natural language descriptions of a narrative scenario of end users, real or potential, using an existing or a future interactive system. STDs may take many forms: use cases, structured scenarios, user stories, and natural language expressions of user actions. As such, these STDs contain useful information for initiating the development life cycle of a user interface of this interactive system. On the one hand, when the end user expresses some interaction through these STDs, user interface requirements can be elicited by deriving model fragments from them: user model, task model, domain model, process model, etc. On the other hand, when the end user refers to any previously used system to feed the requirements, usability problems can be derived from user interfaces critiques: usability problems by interaction object, by dialogue box or window, by entire application. Both approaches feed a bidirectional approach where requirements and usability problems co-exist in the same STD. This article presents how FlowiXML supports the entire approach based on a real-world case study for a distributed system for managing teaching students.*

Keywords. Automated User Interface Generation • Business Modelling • Requirements Elicitation

## 1 Introduction

Scenario-based design (Rosson and Carroll 2009) as well as Participatory Design and other forms of user-centred design initiate typically initiate the development life cycle of the User Interface (UI) of an interactive application using *Scenario Textual Descriptions* (STD), which are general-purpose natural language descriptions of a narrative scenario of end users, real or potential, using an existing or a future interactive system.Such STDs usually consist of informal but structured narrative descriptions of interaction sequences between the users and the interactive system, whether it is existing or envisioned. Scenarios have been proved (Rosson and Carroll 2009) to be a valuable mean to elicit, improve, and validate UI requirements. On the other hand, descriptions of the UI domain itself and the UI requirements are also expressed using conceptual models depicting either static (Tam et al. 1998) or dynamic (Fliedl et al. 2003) aspects of the interactive system. The models resulting from this process are supposed to raise the level of abstraction with respect to the implementation (Tam et al. 1998). The models are frequently expressed in a formal way so as to enable model reasoning. The process which ultimately leads to these descriptions, whether they are informal (such as scenarios) or semi-formal (such as models) is Requirement Engineering (RE) (Haumer et al. 1998). STDs have the advantage to describe UI requirements from captured or imagined user interactions through concrete examples (Garland et al. 2001) of the user carrying out her task. This

* Corresponding author.
E-mail. jguerrero@cs.buap.mx
Note: In a previous work (Lemaigre et al. 2008), we have introduced the process of how to elicit model requirements from textual scenarios. This article generalizes the whole process based on Scenario Textual Descriptions with two original aspects: automatic text interpretation to select elements with CRUDS-based UI to support the handling of selected objects and deriving usability problems from the same source.

form is much more representative and evocative for an end user to validate UI requirements than models that are mainly used by software engineers. Models, e. g., domain models, user models, are expressed in a way that maximizes desirable properties such as completeness, consistency, and correctness (Vanderdonckt 2005). But their expression is significantly less understandable for end users who are often in trouble of validating their UI requirements when they are confronted to models. Consequently, both types of descriptions, scenarios and models, are needed interchangeably in order to conduct a proper process that will effectively and efficiently feed the remainder of the UI development life cycle. STDs could be also effectively used in various configurations of the user interface development life cycle, like forward engineering (Simarro et al. 2005), reverse engineering (Bouillon et al. 2004), adaptation of user interfaces (López-Jaquero et al. 2007), and automated evaluation (Beirekdar et al. 2002).

We introduce model elicitation as the activity of transforming textual scenarios into models that are pertaining to the UI development. The remainder of this article is structured as follows: some related work is reported in Section 2. Three levels of model elicitation are defined in Section 3 and consistently described and discussed in the light of a model elicitation tool implementing these techniques. Section 4 gives another example. Section 5 will sum up the benefits and the shortcomings of the model elicitation techniques investigated so far and will present some future avenues for this work.

## 2 Related Work

Model elicitation consists of transforming scenarios into models so that they are usable in the rest of the UI development life cycle (Haumer et al. 1998), for instance by conducting a model-driven engineering method (e. g., Clerckx et al. 2006; Vanderdonckt 2005). Model verbalization (Jarrar et al. 2014) is the inverse process: it consists of transforming model elements into textual scenarios while preserving some quality properties (e. g., concision, consistency). Any model may be considered for this purpose: models found in HCI (e. g., task, user) or in RE (e. g., domain, organization). In (Bono and Ficorilli 1992), the system restates queries expressed on a domain model (here, an entity-relationship attribute model) into natural language expression. As such, model elicitation is not new in software engineering (Fliedl et al. 2005, 2004), but at least four significant works have been conducted in Human-Computer Interaction (HCI):

1. U-TEL (Lu et al. 1999) is a user-task elicitation software that enables designers to allocate elements of a textual scenarios into elements of three models: actions names (relevant to the task model), user classes (relevant to a user model), and objects names (relevant to a domain model). This allocation can be conducted manually or automatically.
2. In (Caffiau and Portet 2017), a task model is expressed through a STD and a mapping (Simarro et al. 2005), manipulate, and correct better a STD representation of the task model than the task model itself, which may involve a special notation.
3. T2T (Paris et al. 2002) is a tool for automatic acquisition of task elements (names and relationships) from textual documents such as manuals. Another version exists for the same purpose from a domain model (here, an object-oriented diagram) (Lu et al. 1999) and fro multiple heterogeneous sources (Lu et al. 2002).
4. Garland et al. (Garland et al. 2001) present general software for gathering UI requirements from examples containing various elements that are relevant for different models, but models are not constructed per se.

From these works, we observed the following shortcomings: some, e. g., (Garland et al. 2001) do not produce a genuine model in the end, some other produce model elements that are relevant to HCI (e. g., Garland et al. 2001; Paris et al. 2002; Paternò and Mancini 1999), but the only some model elements are derived (e. g., task names)

or they mostly focus on task models whereas several models are typically found in HCI, not only the task model. When other models are considered, e. g., the user and the domain (Lu et al. 1999), only the names of the classes are captured. In this article, we would like to capture all elements (classes, attributes, and relationships) of several interrelated models to inform the UI development life cycle. It is however fundamental that the task model is considered to initiate a full model-driven engineering life cycle (Clerckx et al. 2006; Paternò and Mancini 1999). DYNAMO-AID (Clerckx et al. 2006) provides a distribution manager which distributes the sub-tasks of a task model to various computing platforms in the same physical environment, thus fostering a task-based approach for distributing UIs across locations of the physical environment. In the next section, an elicitation of UI model elements is provided according to three levels of sophistication.

## 3 User Interface Model Elements Elicitation

In order to effectively support UI model elicitation, the model elements that are typically involved in the UI development life cycle should be considered. Figure 1 reproduces a simplified version of the ontology of these model elements that will be used throughout this article: only classes and relationships are depicted here for concision, not their attributes and methods. The complete version of this ontology along with its definition and justification is detailed in (Guerrero Garcia et al. 2008). We choose this ontology because it characterizes the concepts used in the development life cycle of UIs for workflow systems, which are assumed to have the one of the largest coverage possible. Any other similar ontology could be used instead. In this ontology, tasks are organized into processes which are in turn ordered in a Workflow. A job consists of a logical grouping of tasks, as we know them (Paternò and Mancini 1999). Jobs are usually assigned to organizational units (e. g., a department, a service) independently of the workers who are responsible to conduct these jobs. These workers are characterized thanks to the notion of user stereotype. But a same task could require other types of resources such as material resources (e. g., hardware, network) or immaterial resources (e. g., electricity, power). A task may manipulate objects that can invoke methods in order to ensure their role. Figure 1 represents the conceptual coverage of model elements that will be subject to model elicitation techniques. This coverage is therefore larger than merely a task, an object, a user as observed today in the state of the art. In the next subsections, three progressively more sophisticated elicitation techniques based on this ontology will be described, motivated, and exemplified on a running textual scenario. This scenario explains the workflow for obtaining administrative documents in a town hall.

### 3.1 Model Elicitation Level 1: Manual Classification

The UI designer is probably the most reliable person to identify in the textual scenario fragments that need to be elicited into model elements. Therefore, manual classification of model elements remains of high importance for flexibility, reliability, and speed. In a manual classification, any name that represents an instance of a model element belonging to the ontology can be manually selected, highlighted, and assigned to the corresponding concept, such as a task, a job, an organizational unit, etc. Consequently, all occurrences of this instance are automatically identified in the scenario and highlighted in the colour assigned to this concept. For instance, grey for an object, yellow for a user, red for an organizational unit, blue for a task. This colour coding scheme can be parametrized according to the designer's preferences.

Elicitation of a class. Any class belonging to the ontology can be manually classified according to the aforementioned technique. For example, "statement" is considered as an object instance in Figure 2 and is therefore assigned to the corresponding tab . Since a model element may appear in the scenario in multiple alternative forms (e. g.,

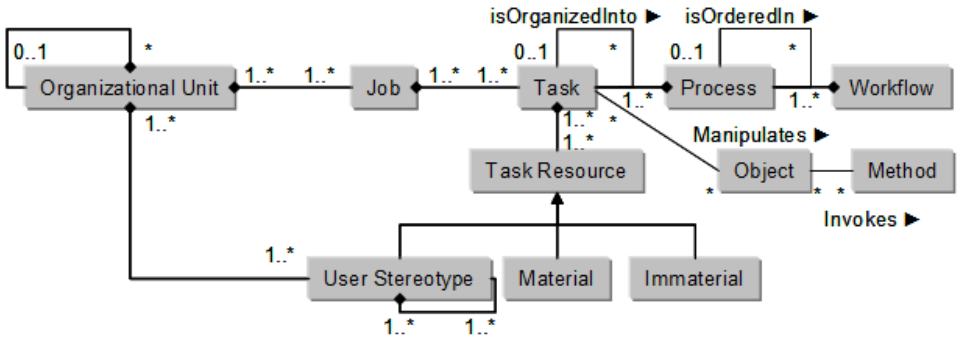



*Figure 1: Simplified ontology of the model elements.*

a plural form, a synonym), an alias mechanism enables designers to defines names that are considered equivalent to a previously defined one. For example, "statements" and "stated text" could be considered aliases of "statement".

In User-Centred Design (UCD), tasks, users, and objects are often considered as first-class citizens. Therefore, it is likely that the design will initiate the classification by identifying firstly tasks and related objects for instance. An object or a task could be of course elicited separately. In order to speed up this activity, the designer may directly associate a task to its related object when selected according to the same mechanism. All occurrences are highlighted similarly. Figure 3 illustrates this situation: a "birth statement" object is selected and a task "issuing" is attached to this object in order to create a complete task "issuing a birth statement".

A special support exists for tasks: at any time, the designer may specify for a task:

- A predefined task type: a taxonomy of task types (e. g., communicate, create, delete, duplicate) is made accessible for the designer to pick a name from, while a definition for each task type is displayed. This taxonomy consists of 15 basic task types that are decomposed into +/- 40 synonyms or sub-task types as used in the UsiXML User Interface Description Language [18]. Each predefined task type comes with a precise definition and scope of the task, some synonyms if any, and its decomposition into sub-tasks if any.
- A custom task name: any non-predefined task name can be entered, such as "issuing" in Figure 3.
- A pattern of tasks: any set of predefined task types and of custom task names. Such a set can be defined by the designer and reused at any time. For example, the pattern CRUD (acronym for Create, Read, Update, Delete) will automatically enter four predefined task types for a designated object.

Elicitation of an attribute. The same technique is used in order to elicit an attribute of a class: either this attribute is predefined in the ontology (e. g., "frequency" to denote the frequency of a task) or a custom name can be manually entered. For example, in Figure 4, the designer has identified in the scenario the expression denoting the frequency of task and therefore elicits this attribute for the corresponding task (here, "ticketing"). The attribute is then represented as a facer of the corresponding task. Figure 5 graphically depicts the three main steps for entering a custom name for an attribute, here an organizational unit. The location of an organization unit is an attribute that does not belong to the ontology. Therefore, once such a parameter has been selected (Figure 5a), it can be identified with a unique name (Figure 5b), and then included in the hierarchy (Figure 5c).

Elicitation of a relationship. By using drag and drop, the designer can arrange model elements

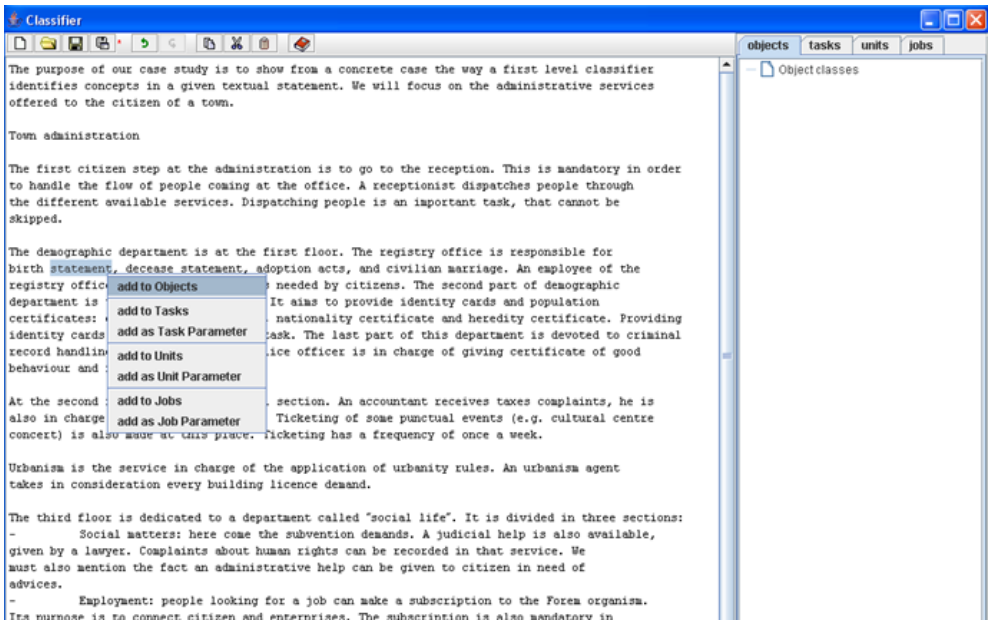



*Figure 2: Elicitation of a class (here, an object) in manual classification.*

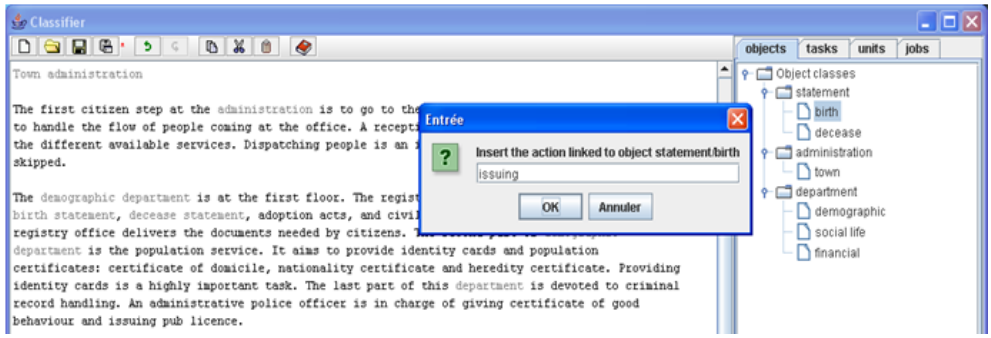



*Figure 3: Assigning a task to a already defined object.*

in their corresponding hierarchy in order to reflect the decomposition relationships of Figure 1. For example, a task is decomposed into sub-task, tasks are composed in a process, processes are composed into a workflow. Apart from these decomposition relationships, only the "manipulates" relationship between a task and an object can be encoded in this level because it can be recorded thanks to the special support for tasks described above. For example in the right pane of Figure 3, the object "statement" is further refined into the two sub-classes "birth statement" and "death statement", that automatically inherit from the attributes of the super-class.

### 3.2 Model Elicitation Level 2: Dictionary-based Classification

The model elicitation technique described in the previous sub-section, although flexible, reliable, and fast, represents a tedious task for the designer since it is likely to be repeated. Therefore, instead of manually designating in the textual scenario the fragments that are subject to model elicitation, these fragments could be automatically classified according to a dictionary of model terms. We distinguish two categories of dictionary:

1. Generic dictionaries contain fragments representing model elements that are supposed to

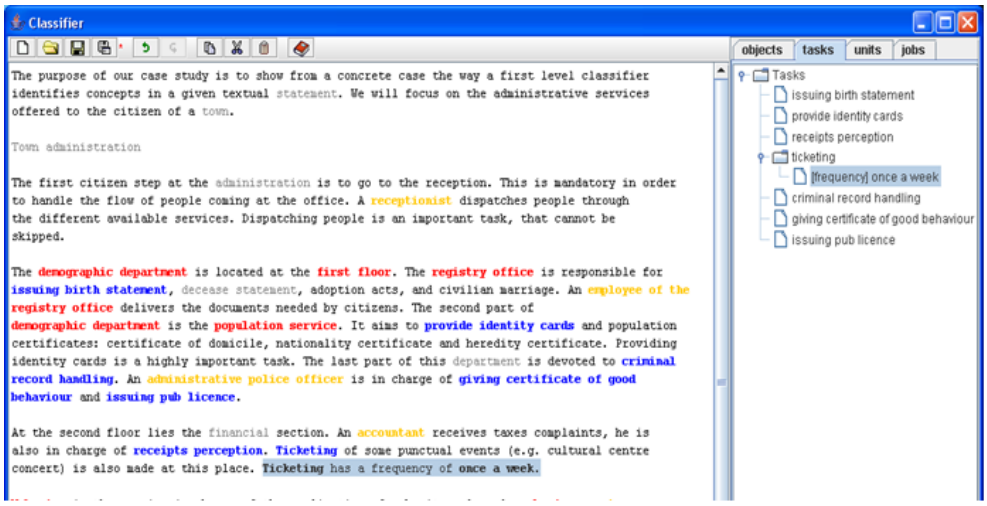



*Figure 4: Elicitation of a predefined attribute for a task.*

be domain-independent (e. g., "a worker", "a manager", "a clerk" for a user model; "create", "read", "update", "delete" for a task model, etc.)

2. Specific dictionaries that contain fragments representing model elements that are domain-dependent (e. g., a "physician", "a pharmacist" in medicine for a user model; "administrate" for a task model, "physiology " for a domain model).

Each dictionary may contain predefined terms (like the task types) and aliases (e. g. plural, synonyms) in order to maximize the coverage of the automatic classification. In order to tailor this classification, two types of filters could be applied Tam et al. 1998:

1. Positive filters force some model terms to be considered anyway, whatever the domain or the context of use are.
2. Negative filters prevent the automatic classification from classifying irrelevant terms, such as articles (e. g., "the", "a"), conjunctions (e. g., "with", "along").

The terms collected in such filters can be edited manually within any ASCII-compliant text editor. The advantage of this dictionary-based classification over the manual one is certainly its speed: in a very short amount of time, most terms belonging to the dictionaries, modulo their inclusion or rejection through the usage of filters, are classified. The most important drawback of this technique is that the identified terms are not necessarily located in the right place in their corresponding hierarchies. For example, a task hierarchy resulting from this process may consist of a one-level hierarchy of dozens of sub-tasks located in the same level without any relationships between them. In order to overcome this serious drawback, a semantic-based technique is required that is addressed in the next subsection.

### 3.3 Model Elicitation Level 3: Towards Semantic Understanding

Different techniques exist that elicit model elements from textual scenarios, but so far they have never been applied in HCI to our knowledge: syntactic tagging (Fliedl et al. 2003), semantic tagging (Fliedl et al. 2004), chunk parsing (Fliedl et al. 2004). Genuine semantic understanding requires natural language understanding and processing, which is far beyond the scope of this work. What can be done however is to substitute a semantic understanding by a combination of syntactic and semantic tagging (Fliedl et al. 2005, 2004) in order to recognize possible terms that express, depict, reveal model elements. For instance, a

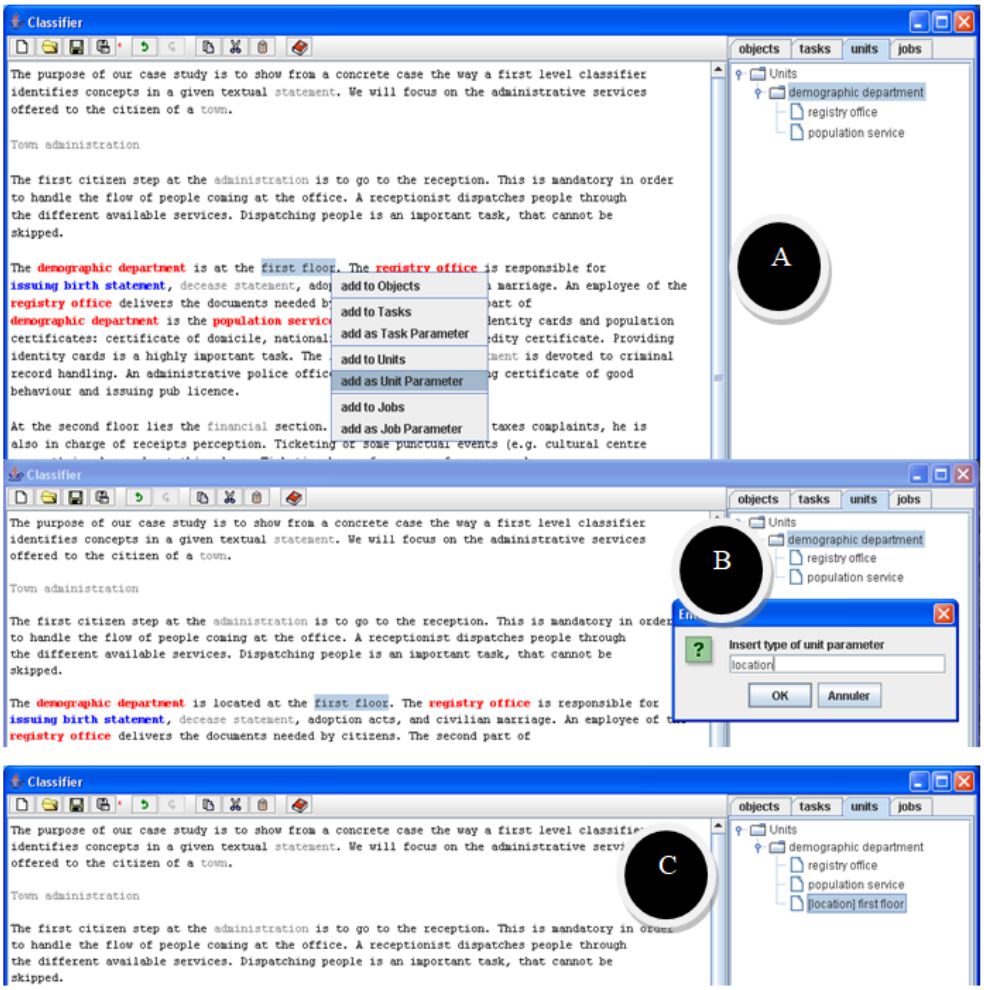



*Figure 5: Section a. Elicitation of a custom attribute for an organizational unit: selection. Selection b Elicitation of a custom attribute for an organizational unit: identification. Section c. Elicitation of a custom attribute for an organizational unit: inclusion.*

scenario sentence like “An accountant receives taxes complaints, but she is also in charge of receipts perception” should generate: a task “Receive taxes complaint”, a task “charge of receipts perception”, both being assigned to the user stereotype “Accountant”, and a concurrency temporal operator between those two tasks because no specific term is included to designate how these tasks are actually carried out by an accountant. We may then assume the most general temporal operator, like a concurrency temporal operator. To reach this goal, this level attempts to identify possible terms in a syntactical structure (e. g., a set, a list, a sequence) that depicts a pattern for inferring for instance a task, another task with a temporal constraint, etc. For each model element, a table of possible terms involved in this pattern structure is maintained in accordance with the semantics defined in Figure 1. On the one hand, this pattern matching scheme is syntactical because it is only based on detecting a particular combination of terms. On the other hand, those terms are assumed to reflect the precise semantics defined in the ontology. But we cannot say that this is a true semantic understanding anyway. Table 1 shows some excerpts of possible terms related to the concept

of task, along with its attributes, while Table 2 shows some possible terms for detecting possible temporal relationships between tasks. This pattern matching can be executed automatically or under the designer's control who validates each matching one by one. The reserved names for model elements (e. g., task, the task attributes, and the temporal operators between the tasks) are read from the XML schema definition of the underlying User Interface Description Language (UIDL), which is UsiXML (Vanderdonckt 2005) in this case. After performing the elicitation of model elements according to any of the three aforementioned technique, the model elicitation tool can export the results in UsiXML files for the whole set of models or for any particular combination (e. g., only the tasks with the users) at any time. Afterwards, this file can be imported in any other UsiXML-compliant editor, such as IDEALXML for graphical editing.

## 4 Conclusion

In this article, we have investigated three different techniques for eliciting model elements from fragments found in a textual scenario. These three techniques are progressively more advanced in terms of consideration of the possible terms found in the scenario: from purely manual syntactical classification until ontology-based pseudo-semantic understanding. Beyond the facilities for automated classification of terms into the respective models, that are compatible with the initial ontology, the model elicitation tool allows editing facilities within a same model and across models. Its main drawback today is the lack of graphical visualization of inter-model relationships or intra-model relationships, others than merely decomposition relationships. For the moment, these relationships are only collected in a table that can be further edited. In the near future, we would like to refine the level 3-technique in terms of possible combinations of terms in an expression to be subject for the pattern matching.

## References

Beirekdar A., Vanderdonckt J., Noirhomme-Fraiture M. (2002) A Framework and a Language for Usability Automatic Evaluation of Web Sites by Static Analysis of HTML Source Code. In: Kolski C., Vanderdonckt J. (eds.) Computer-Aided Design of User Interfaces III, Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces, May, 15-17, 2002, Valenciennes, France. Kluwer, pp. 337–348

Bono G., Ficorilli P. (1992) Natural language restatement of queries expressed in a graphical language. In: Entity-Relationship Approach—ER'92, pp. 357–374

Bouillon L., Vanderdonckt J., Chow K. C. (2004) Flexible re-engineering of web sites. In: Vanderdonckt J., Nunes N. J., Rich C. (eds.) Proceedings of the 9th International Conference on Intelligent User Interfaces, IUI 2004, Funchal, Madeira, Portugal, January 13-16, 2004. ACM, pp. 132–139 http://doi.acm.org/10.1145/964442.964468

Caffiau S., Portet F. (2017) La génération automatique de textes comme support de la compréhension de modèle de tâches en conception: une étude préliminaire. In: Proceedings of IHM2017, pp. 125–135 https://hal.archives-ouvertes.fr/hal-01578499/file/1030.pdf

Clerckx T., Vandervelpen C., Luyten K., Coninx K. (2006) A task-driven user interface architecture for ambient intelligent environments. In: Proceedings of the 11th international conference on Intelligent user interfaces. ACM, pp. 309–311

Fliedl G., Kop C., Mayr H. C. (2003) From scenarios to KCPM dynamic schemas: aspects of automatic mapping. In: NLDB, pp. 91–105

Fliedl G., Kop C., Mayr H. C. (2005) From textual scenarios to a conceptual schema. In: Data & Knowledge Engineering 55(1), pp. 20–37

Fliedl G., Kop C., Mayr H., Winkler C., Weber G., Salbrechter A. (2004) Semantic tagging and chunk-parsing in dynamic modeling. In: Natural Language Processing and Information Systems, pp. 440–446

Garland A., Ryall K., Rich C. (2001) Learning hierarchical task models by defining and refining examples. In: Proceedings of the 1st international conference on Knowledge capture. ACM, pp. 44–51

Guerrero Garcia J., Vanderdonckt J., Gonzalez Calleros J. M. (2008) FlowiXML: a step towards designing workflow management systems. In: International Journal of Web Engineering and Technology 4(2), pp. 163–182

Haumer P., Pohl K., Weidenhaupt K. (1998) Requirements elicitation and validation with real world scenes. In: IEEE Transactions on Software Engineering 24(12), pp. 1036–1054

Jarrar M., Keet C. M., Dongilli P. (2014) Multilingual verbalization of ORM conceptual models and axiomatized ontologies. In:

Lemaigre C., Guerrero J., Vanderdonckt J. (2008) Interface model elicitation from textual scenarios. In: Human-Computer Interaction Symposium. Springer, pp. 53–66

López-Jaquero V., Vanderdonckt J., Simarro F. M., González P. (2007) Towards an Extended Model of User Interface Adaptation: The Isatine Framework. In: Gulliksen J., Harning M. B., Palanque P. A., van der Veer G. C., Wesson J. (eds.) Engineering Interactive Systems - EIS 2007 Joint Working Conferences, EHCI 2007, DSV-IS 2007, HCSE 2007, Salamanca, Spain, March 22-24, 2007. Selected Papers. Lecture Notes in Computer Science Vol. 4940. Springer, pp. 374–392 https://doi.org/10.1007/978-3-540-92698-6_23

Lu S., Paris C., Vander Linden K. (1999) Toward the automatic construction of task models from object-oriented diagrams. In: Engineering for human-computer interaction. Springer, pp. 169–189

Lu S., Paris C., Vander Linden K. (2002) Computer Aided Task Model Acquisition From Heterogeneous Sources. In: Proc. of 5th Asia Pacific Conference on Computer Human Interaction APCHI, pp. 878–886

Paris C., Linden K. V., Lu S. (2002) Automated knowledge acquisition for instructional text generation. In: Proceedings of the 20th annual international conference on Computer documentation. ACM, pp. 142–151

Paternò F., Mancini C. (1999) Developing task models from informal scenarios. In: CHI'99 Extended Abstracts on Human Factors in Computing Systems. ACM, pp. 228–229

Rosson M. B., Carroll J. M. (2009) Scenario based design. In: Human-computer interaction. Boca Raton, FL, pp. 145–162

Simarro F. M., López-Jaquero V., Vanderdonckt J., González P., Lozano M. D., Limbourg Q. (2005) Solving the Mapping Problem in User Interface Design by Seamless Integration in IdealXML. In: Gilroy S. W., Harrison M. D. (eds.) Interactive Systems, Design, Specification, and Verification, 12th International Workshop, DSVIS 2005, Newcastle upon Tyne, UK, July 13-15, 2005, Revised Papers. Lecture Notes in Computer Science Vol. 3941. Springer, pp. 161–172 https://doi.org/10.1007/11752707_14

Tam R., Maulsby D., Puerta A. R. (1998) U-TEL: A tool for eliciting user task models from domain experts. In: Proceedings of the 3rd international conference on Intelligent user interfaces. ACM, pp. 77–80

Vanderdonckt J. (2005) A MDA-compliant environment for developing user interfaces of information systems. In: International Conference on Advanced Information Systems Engineering. Springer, pp. 16–31

# Model-Driven Time-Series Analytics

Sabine Wolny[*,a], Alexandra Mazak[a], Manuel Wimmer[a], Rafael Konlechner[a], Gerti Kappel[a]

[a] CDL-MINT, TU Wien, Austria

Abstract. *Tackling the challenge of managing the full life-cycle of systems requires a well-defined mix of approaches. While in the early phases model-driven approaches are frequently used to design systems, in the later phases data-driven approaches are used to reason on different key performance indicators of systems under operation. This immediately poses the question how operational data can be mapped back to design models to evaluate existing designs and to reason about future re-designs. In this paper, we present a novel approach for harmonizing model-driven and data-driven approaches. In particular, we introduce an architecture for time-series data management to analyse runtime properties of systems which is derived from design models. Having this systematic generation of time-series data management opens the door to analyse data through design models. We show how such data analytics is specified for modelling languages using standard metamodelling techniques and technologies.*

Keywords. Model-Driven Engineering • Time-Series • Data Analytics • Language Engineering

## 1 Introduction

In model-driven engineering (MDE), models are the central artefact and used as a main driver throughout the software development process, finally leading to an automated generation of software systems (Lara et al. 2015). In the current state-of-practice in MDE (Brambilla et al. 2017; Karagiannis et al. 2016), models are used as an abstraction and generalization of a system to be developed. By definition, a model never describes reality in its entirety, rather it describes a scope of reality for a certain purpose in a given context (Brambilla et al. 2017). Thus, models are mostly used as *prescriptive models* for creating a software system (Heldal et al. 2016). Such design models determine the scope and details of a domain of interest to be studied. For this purpose, different types of general modelling languages (e. g., state charts, class diagrams, etc.) may be used or domain-specific languages (DSLs) (Karagiannis et al. 2016) may be employed. It has to be emphasized that engineers typically have the desirable behaviour in mind when creating a system, since they are not aware in these early phases of many deviations that may take place at runtime (van der Aalst 2016).

According to Brambilla et al. (2017) the implementation phase deals with the mapping of prescriptive models to some executable systems and consists of three levels: (*i*) the *modelling level* where the models are defined, (*ii*) the *realization level* where the solutions are implemented through artefacts that are used in the running system, and (*iii*) the *automation level* where mappings from the modelling to the realization phase are made. However, these levels are currently only used for down-stream processes. The possibility of up-stream processes is mostly neglected in MDE (Mazak and Wimmer 2016). Especially, for later phases of the system lifecycle *descriptive*

* Corresponding author.
E-mail. wolny@big.tuwien.ac.at
This work has been supported by the National Foundation for Research, Technology and Development (CDG), the Austrian Federal Ministry of Science, Research, and Economy (BMWFW Austria), and the TU Wien research funds.

*models* may be employed to better understand how the system is actually realized and how it is operating in a certain environment (Mazak and Wimmer 2016). Compared to prescriptive models, those descriptive models are only marginal explored in the field of MDE, and if used at all, they are built manually.

In this paper, we move towards a well-defined mix of approaches to better manage the full lifecycle of systems by combining prescriptive and descriptive model types. In particular, we introduce a model-driven time-series data analytics architecture for harmonizing model-driven and data-driven approaches. Based on this architecture, we show how data analytics can be specified for modelling languages using standard metamodelling techniques. This means, design-oriented languages are equipped with extensions for representing runtime states as well as runtime histories, which in turn allow the formulation and computation of runtime properties with the Object Constraint Language (OCL). This approach has the advantage to directly interpret measurements and events within the design models without introducing an impedance mismatch.

The remainder of this paper is structured as follows. Section 2 provides the background for this paper by introducing a motivating example which is subsequently used as running example. Section 3 gives an overview of our architecture for unifying model-driven and data-driven approaches. In Section 4, we present in detail how time-series analytics can be integrated in metamodels. Section 5 discusses the related work. Finally, in Section 6, we conclude with an outlook on future work.

## 2 Motivating Example

In this section, we introduce a motivating example, which will subsequently become the running example of this paper. We first describe the example from the modelling perspective, then from the realization perspective with a focus on runtime data collection, and finally conclude with the challenges we aim to address with this paper.

**Model-Driven Perspective**

As our motivating example, we consider a grip-arm robot (gripper) with different position properties of axis angles: `BasePosition (BP)`, `MainArmPosition (MAP)`, and `GripperPosition (GP)`. From a device point of view (cf. Figure 1(a)), the structure of the gripper component and its behaviour are modelled at design time by a subset of a SysML-like language, i.e., blocks with associated state machines. The top of Figure 1(a) shows the specific properties (`BP`,`MAP`,`GP`) of the block, whereas the actual property values depend on the different states (e.g., `Idle`, `Pick Up`). The states are given at the bottom of Figure 1(a).

By the given state machine, property value changes are modelled. The gripper has certain positions at initialization, in state `Idle` and in state `Pick Up`. The assumption of the modelled state machine is that as soon these states are reached, the position values are set. However, such state machines are a kind of black box, where only the discrete values before entering the state and after leaving the state are known (cf. Figure 1(b)). While this may be sufficient for several design tasks and discrete systems, for continuous systems more information may be required. This is in particular true for our example case. The gripper represents a continuous system, since it does not immediately realize the next position, but needs time to move to the given place. Usually, such information is not directly given in a design model, but it may be important for several tasks such as optimization, validation, and verification. The ability to observe property value changes over time within states may contribute to capture the current capabilities and shortcomings of systems. Thus, the presented approach of this paper aims to transform the black box into a so-called "grey box" to make the effects of value changes visible (see Figure 1(c)). For instance, observation sequences of property value changes are an important base information of a system's operation to compute operating figures to check if the behaviour of each

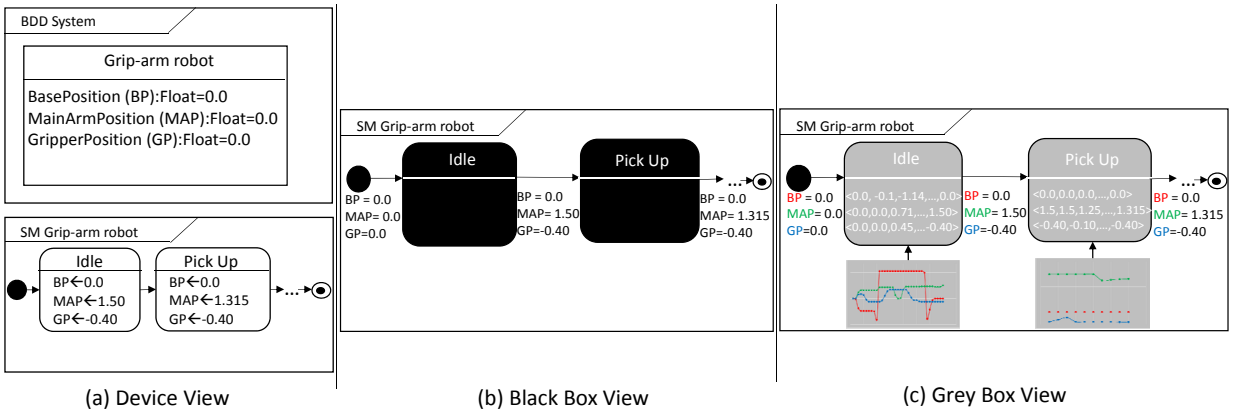



*Figure 1: Different model-based views on a grip-arm robot.*

gripper's axis corresponds to the defined one in the design model.

**Data-Driven Perspective**

For the technical realization of our example, we developed a simulation model of the gripper consisting of three angle sensors, which we executed by the open source tool Blender[1]. We deploy the scenario of a pick-and-place unit, where the gripper picks up different color-coded work pieces, place them on a test rig, picks the items up again and puts them down, depending on their red or green color, in two different storage boxes. The simulation environment receives its commands via Message Queue Telemetry Transport (MQTT) from a server controller implemented with Kotlin[2]. The simulation enables to acquire transient data streams in real-time from the angle axes of the gripper (BP, MAP, GP, unit is radian), which are equipped with sensors.

To react on events of interest provided by these data streams, we employ the publish/subscribe pattern. In our example, we subscribe to the sensor topic to receive in a temporal distance of 15 milliseconds the filtered data streams of the sensors of the gripper during simulation. Thereby, we are interested in property value changes (i. e., positions of the axes) in the simulation at given points in time. Messages from the sensor topic are defined in JSON[3] specifying the sending unit as well as the measured data. The following example shows such a message from the angle sensors of the gripper to the controller.

```
{"entity": "GripperArm",
"basePosition": 0.0,
"mainArmPosition": 0.0,
"gripperPosition": 0.0}
```

This example shows the positions of the angle axes at system initialization (see Figure 1). The angle position of each axis has the value 0.0. The default range of the angle values is $[-\pi,\pi]$. To analyze our scenario, it is important to save the measured data over time. For this purpose, we use the time series database InfluxDB[4]. InfluxDB allows us to store a large amount of time-stamped data. In addition, by the tool Grafana[5] we can visualize our stored sensor values.

**Challenges**

Our motivating example is discussed from two angles: (*i*) from the model-driven, i. e., how the intended system should work, and (*ii*) from the data-driven, i. e., how measurements can be taken from the running system to reason about the actual realization. While the first perspective is lacking concepts to define runtime data such as

[1] https://www.blender.org
[2] https://kotlinlang.org
[3] http://json.org/example.html
[4] https://www.influxdata.com
[5] https://grafana.com

time-series, the second perspective has to correctly interpret the collected measurements. The challenge is how to overcome the gap between those two perspectives (*i*) to monitor important data from operation, (*ii*) to align the measurements with the design model in order to provide a semantic anchoring of the data, and (*iii*) to provide meaningful analytics whereas the results of the analytics are interpretable for the given design models to reason about improvements or fulfilments of given requirements.

## 3 Unifying Architecture for Model-Driven and Data-Driven Approaches

In order to allow a smooth integration of model-driven and data-driven approaches, we present in this section an architecture, which builds on the classical model to system downstream in terms of code generators, but at the same time, supports an upstream in terms of mapping data back to design models. Figure 2 gives an overview of this architecture. In the following section, a more detailed description of the different parts will be presented based on our running example.

The proposed architecture consists of four main parts. First, the left hand side of Figure 2 captures the classical downstream MDE approach (cf. (a) in Figure 2). At the metamodel layer, the `design language` is defined with the help of a metamodelling language (in our setting Ecore). Conforming to the design language, the `design models` are defined at the model level describing the static (i. e., structure) and dynamic aspects (i. e., behaviour) of a system to be developed. For the vertical transition from the modelling to the realization level we assume the existence of model-to-text transformations for code generation. Thus, this part of our architecture describes how we can derive the executable system from the design model as is the state-of-the-art in MDE.

Second, we continue with defining the first part of the up-stream process of runtime data to the design model (cf. (b) in Figure 2). In addition to the actual systems, the `runtime observer` is generated out of the design model. The runtime observer collects important information from the running system to represent the current state of the system. Those observations should not only be recorded by observing the running system, but should be also representable at the model level. Thus, we extend the `design language` with a dedicated `runtime language`. This metamodel defines the syntax to represent snapshots of the running system connected to the design model elements. Those snapshots are represented in the `runtime state models` which extend the `design models` and may be directly updated by the `runtime observer` during runtime. In summary, this part of our architecture maps runtime data at the model level for one single point in time and may be used to monitor a system on the model level.

Third, we define the runtime history of a system (cf. (c) in Figure 2). For reasoning about, e.g., property value distributions, it is important to have the complete history of value changes as starting point as one snapshot is definitely not sufficient for such computations. Thus, in the `time series database` the observations of the running system are stored. Based on these collected observations, the `runtime history models` may be directly updated. These models conform to the `runtime history language`, which is an extension of the `runtime language`. In the `runtime history language`, the syntax is defined for representing histories of runtime phenomena of interest, e.g., property values, events, etc.

Finally, after defining those concepts for storing observation histories at the model level, it is also possible to analyse the stored observations (cf. (d) in Figure 2). For this purpose, we define `runtime properties` based on the Object Constraint Language (OCL) by introducing derived properties for the metamodel elements. These derived properties enable us, e.g., (*i*) to compute descriptive statistics, (*ii*) to evaluate monotony behaviour of value changes, or (*iii*) to compute lower and upper bounds of properties to mention just a few examples. Based on the `runtime properties`, the `runtime property values` are computed by analysing the collected time-series. Thus, runtime

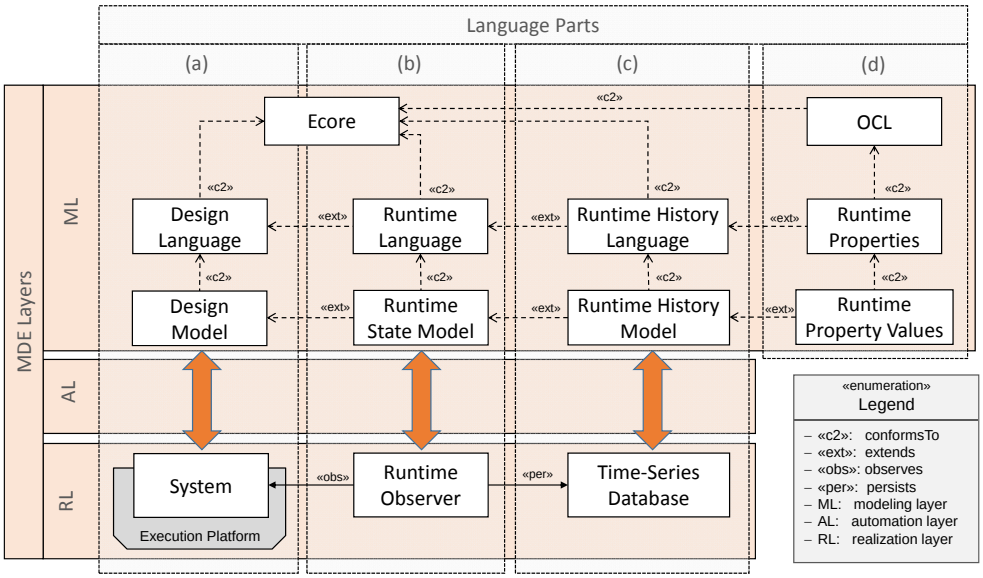



*Figure 2: Unifying architecture for model-driven and data-driven perspectives.*

data is back propagated to the design models and this mapping allows to interpret the data through the design model elements as there is a clear traceability guaranteed from design elements, runtime states, and runtime histories.

By this architecture, we are able to harmonize model-driven and data-driven approaches, where time-series data management of systems at runtime can be derived from initial design models and be used again at the model layer by importing the time-series to model structures. How this model structures are defined is the topic of the next section.

## 4 Metamodelling Blueprint for Enhancing Models with Time-Series Analysis

Based on our running example, we further detail in this section how the afore presented architecture can be realized for a given language. In particular, we show for the introduced design modelling language, how the extensions for runtime states, runtime histories, and runtime properties are defined as reusable metamodelling blueprints. The time-series analysis we are focusing on for demonstration purposes is about property value changes of the axis angles (i. e., BP, MAP, GP) of the gripper in our running example.

### Design Elements

As already mentioned before, our starting point is the availability of a design modelling language expressed in Ecore. For our running example, we model the structure of the gripper with its properties as a kind of block diagram similar to what is known from SysML. A block has an associated state machine, where different states and transitions are defined. For states, assignments can be defined, which are executed when a state is activated. The assignments in our exemplary language are simple value assignments for the properties of a block. The resulting metamodel for the described design language is shown in Figure 3.

### Runtime States

In order to express concrete runtime states on the model level, the metamodel has to be extended with runtime concepts. For this task, there are several existing approaches available, e. g., (Engels et al. 2000; Mayerhofer et al. 2013; Meyers et al. 2014). Most of them add additional metamodel elements to the design language to describe what

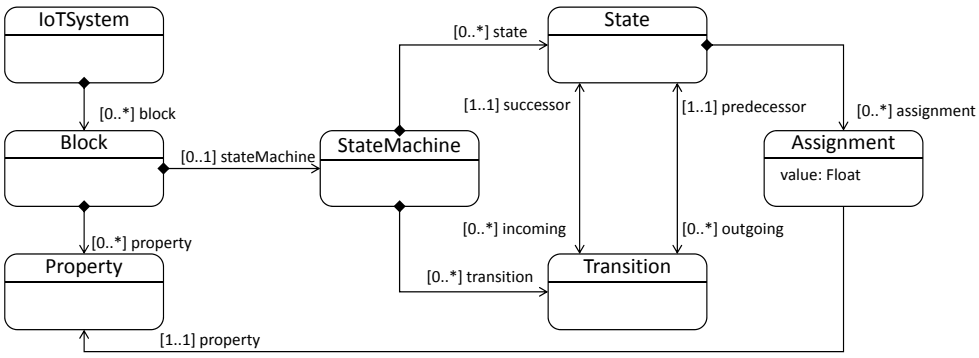



*Figure 3: Design metamodel for the running example.*

runtime phenomena are of interest and how they are connected to design concepts. For our running example, the `runtime language` is considered as an extension of the design metamodel to allow representing property values for a given point in time (i. e., for a snapshot of the running system). In addition, transitions may fire during runtime. Thus, the concept of transition firing is introduced. While values are considered by measurements during the operation phase, the firing of transitions are categorized as events. Please note that the relation to the design concepts has to be clearly stated by the runtime concepts, e. g., the value concept is related to the property concept. Figure 4 captures the concrete realization of the runtime extension for our design language.

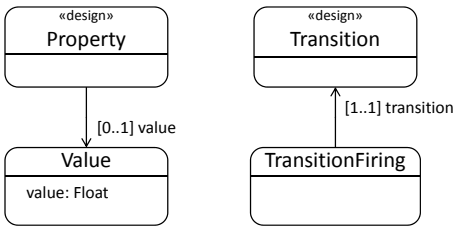



*Figure 4: Runtime metamodel for the running example.*

## Runtime Histories

To reason about operation figures going beyond one snapshot in time such as distributions, upper and lower bounds, histories of property values and event sequences are necessary. Therefore, we need another extension which allows to represent the runtime history of a system. For this, we introduce a novel metamodelling blueprint which introduces the concept of history by providing a sequence of steps having a particular timestamp associated. Figure 5 illustrates the separation of steps into measurement snapshots and event snapshots. These specific steps are forming the event histories and measurement histories. The measurement history contains all measurement snapshots, which comprise values for given time steps. Event histories do the same for events. In our running example, the measurement snapshots refer to the value runtime concept introduced by the runtime extension and the event snapshots are referring to the transition firing concept.

Having this base structure introduced allows us to represent time-series data in design models by using runtime concepts as glue between models and data.

## Runtime Properties

For analysing the time-series data represented in the aforementioned runtime history models, we introduce derived properties which actually represent runtime properties. Derived properties have been already used heavily in the past for deriving additional information from given structures and values. As we explicitly represent runtime histories as model structures, we can make use of derived properties to derive runtime information from the base time-series recorded during operation.

In the following we state three runtime properties for the given design language, namely for

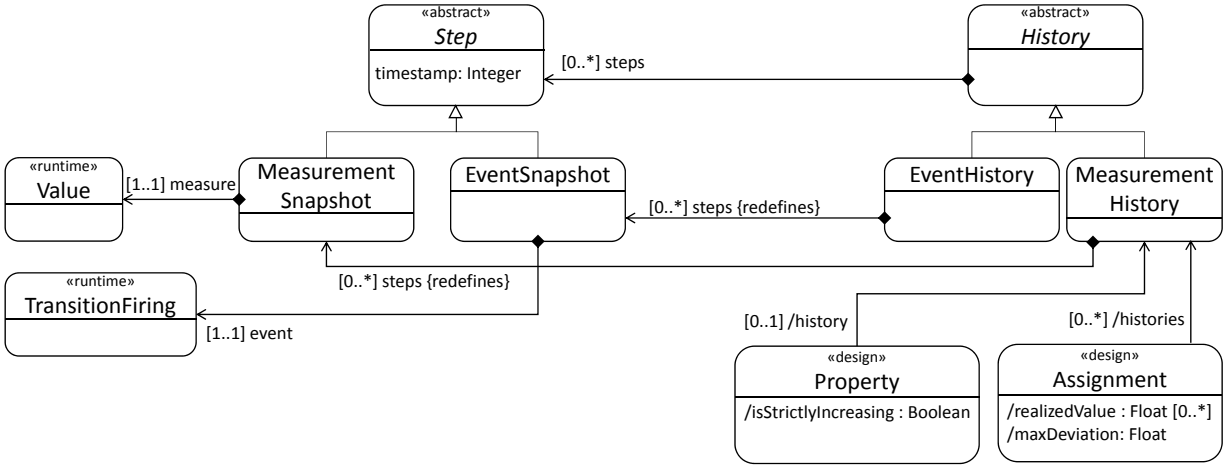



*Figure 5: Runtime history metamodel for the running example.*

the `Property` metaclass and the `Assignment` metaclass. We use standard OCL to derive the runtime properties.

For properties defined in blocks, it may be of interest if their values are strictly increasing over time or not. This can be expressed in OCL by providing a derived history reference for properties from the complete measurement history. The reference only contains the slice of the full history which concerns the given property. Using this reference, we can simply collect all values as a sequence (the ordering expresses the occurrence of the values). If the sorted sequence corresponds to the base sequence, then the property is strictly increasing.

```
context Property::isStrictlyIncreasing:
    ↪Boolean
derive: self.history.steps.measure.value->
    ↪flatten()->sortedBy(x|x) = self.
    ↪history.steps.measure.value->flatten()
```

Concerning the assignments within states, one may be interested if the stated value is actually realized by the system. For this, the realized values may be collected by taking the last snapshots of all assignment executions for a given assignment.

```
context Assignment::realizedValues:Set(Float
    ↪)
derive: self.histories -> collect(x|x.steps
    ↪-> last()) -> collect(x|x.measure.
    ↪value) -> asSet()
```

Having the set of realized values, the maximum deviation is calculated by introducing another derived property which builds on the previous one.

```
context Assignment::maxDeviation:Float
derive: self.realizedValues -> collect(x|(
    ↪self.value-x).abs()) -> sortedBy(x|x)
    ↪-> last()
```

## 5 Related Work

In this section, we discuss existing work with respect to the contribution of this paper, namely the combination of model-driven and data-driven approaches with a focus on time-series analytics. Therefore, we first discuss data-driven approaches for enhancing existing domain specific languages (DSLs), and subsequently, we enumerate existing work which proposes dedicated DSLs for time-series analytics.

**Data-driven approaches for DSLs**
An emerging field for data-enhanced modelling languages is Web engineering. For instance, Bernaschina et al. (2017) point to the fact that there is the need for merging Web site navigation statistics of user behaviour with the structure of

the Web application models. The authors show the advantages of combining user interaction models with user tracking information in form of user navigation logs, and details about the visualized content in the pages. Their approach interweaves design time information and runtime execution data of Web sites in order to significantly improve the analysis of user behaviour. In (Artner et al. 2017), we combined navigation models with Markov chains for representing navigation path probabilities, which are derived from execution logs. While these existing approaches for Web applications follow the general idea of combining data-driven and model-driven approaches, the approach of this paper is independent from the actual domain and may be also used in the future to reproduce these existing specific approaches.

Another very active research field is process mining (van der Aalst 2016) which aims to discover process models from workflow execution logs. A variety of process mining algorithms exists that allows the discovery of different process models in different formalisms. In (Wolny et al. 2017), we present an initial architecture how process mining may be related with time-series mining. By this, not only the dependencies between different process steps may be uncovered, but also dependencies between data and process steps are approachable.

Finally, in (Hartmann et al. 2017) the authors present a DSL which allows not only the specification of structural aspects of a systems, but also the definition of so-called learned properties. Such properties are computed from runtime data by using some kind of machine learning algorithms. Our approach directly allows to encode such properties as derived properties based on time-series data computed with OCL as we model the runtime history explicitly. In future work, it will be interesting to combine our time-series analysis with machine learning algorithms as proposed by Hartmann et al. (2017).

**DSLs for Time-Series Analytics**

The OMS3 modelling framework[6] introduces an extensible and lightweight layer for a simulation description expressed as Simulation DSL by using Groovy[7] as a framework for providing the code generator implementation. In (David et al. 2012), the authors present DSL variants in OMS3, e. g., the Ensemble Streamflow Prediction (ESP) DSL. This DSL uses time-series of historic meteorological data as model input to predict future conditions. In their approach, DSLs are employed for time-series unlike in our approach, where we use time-series for domain-specific modelling.

Gekko[8] is an open-source modelling approach for time-series data management and for solving and analysing large-scale time-series models. Gekko may be considered as a kind of DSL with a time-series domain focus. It provides interfaces to statistic packages such as R. In our approach, we use an open-source time-series database which offers besides high-availability storage and monitoring of time-series, application metrics and real-time analytics in addition. Nevertheless, in future work it is of interest to evaluate different possibilities to perform time-series analytics in addition to our current approach.

## 6 Conclusion and Future Work

In this paper, we have introduced an unifying architecture for combining model-driven and data-driven approaches for system engineering. By this architecture, we allow for specifying and computing runtime properties based on time-series data through design models. The extensions needed on the metamodel level are non-intrusive and connected to existing approaches for specifying the operational semantics of languages. The presented runtime history metamodel fragments are applicable for any design modelling language comprising features to be measured and events to be tracked as the current metamodelling languages Ecore and OCL are reused. We demonstrated our

[6] https://alm.engr.colostate.edu/cb/project/oms
[7] http://groovy-lang.org
[8] http://t-t.dk/gekko

approach for a cyber-physical production system case. We have also realized a prototype in Eclipse supporting our approach which is available on our project website[9] .

While the presented approach opens the door for using time-series analytics in a model-driven engineering toolbox, there are still several challenges to be tackled in the future. In particular, we consider the following points on our roadmap: *scalability* (e. g., should the analysis be performed on the model level or directly in the time-series database?), *expressivity* (e. g., which extensions of OCL are necessary for statistical reasoning?), *understandability* (e. g., how to visualize time-series oriented information in diagrams?), and *predictability* (e. g., how to derive and use operations from time-series for predicting future runtime states?).

## References

Artner J., Mazak A., Wimmer M. (2017) Towards Stochastic Performance Models for Web 2.0 Applications. In: Proceedings of the 17th International Conference on Web Engineering (ICWE). Springer, pp. 360–369

Bernaschina C., Brambilla M., Mauri A., Umuhoza E. (2017) A Big Data Analysis Framework for Model-Based Web User Behavior Analytics. In: Proceedings of the 17th International Conference on Web Engineering (ICWE). Springer, pp. 98–114

Brambilla M., Cabot J., Wimmer M. (2017) Model-Driven Software Engineering in Practice. Morgan & Claypool

David O., Lloyd W., II J. C. A., Green T. R., Olson K., Leavesley G. H., Carlson J. (2012) Domain Specific Languages for Modeling and Simulation: Use Case OMS3. In: Proceedings of the International Congress on Environmental Modelling and Software Managing Resources of a Limited Planet, pp. 1201–1207

[9] https://cdl-mint.big.tuwien.ac.at/case-study-artefacts-for-emisa-2017

Engels G., Hausmann J. H., Heckel R., Sauer S. (2000) Dynamic Meta Modeling: A Graphical Approach to the Operational Semantics of Behavioral Diagrams in UML. In: Proceedings of the 3rd International Conference on the Unified Modeling Language (UML). Springer, pp. 323–337

Hartmann T., Moawad A., Fouquet F., Le Traon Y. (2017) The next evolution of MDE: a seamless integration of machine learning into domain modeling. In: Software & Systems Modeling

Heldal R., Pelliccione P., Eliasson U., Lantz J., Derehag J., Whittle J. (2016) Descriptive vs prescriptive models in industry. In: Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems (MODELS). ACM, pp. 216–226

Karagiannis D., Mayr H. C., Mylopoulos J. (2016) Domain-Specific Conceptual Modeling, Concepts, Methods and Tools. Springer

de Lara J., Guerra E., Cuadrado J. S. (2015) Model-driven engineering with domain-specific meta-modelling languages. In: Software and System Modeling 14(1), pp. 429–459

Mayerhofer T., Langer P., Wimmer M., Kappel G. (2013) xMOF: Executable DSMLs Based on fUML. In: Proceedings of the 6th International Conference on Software Language Engineering (SLE), pp. 56–75

Mazak A., Wimmer M. (2016) Towards Liquid Models: An Evolutionary Modeling Approach. In: Proceedings of the 18th IEEE Conference on Business Informatics (CBI). IEEE, pp. 104–112

Meyers B., Deshayes R., Lucio L., Syriani E., Vangheluwe H., Wimmer M. (2014) ProMoBox: A Framework for Generating Domain-Specific Property Languages. In: Proceedings of the 7th International Conference on Software Language Engineering (SLE). Springer, pp. 1–20

van der Aalst W. M. P. (2016) Process Mining - Data Science in Action. Springer

Wolny S., Mazak A., Konlechner R., Wimmer M. (2017) Towards Continuous Behavior Mining. In: Proceedings of the 7th International Symposium on Data-driven Process Discovery and Analysis (SIMPDA), pp. 149–150

# A Practice-Proven Reference Architecture for Model-Based Collaborative Information Systems

Adrian Hernandez-Mendez[*,a], Felix Michel[a], Florian Matthes[a]

[a] Software Engineering for Business Information Systems, Technische Universität München, Germany

Abstract. *This article provides a condensed overview of a layered and model-driven system architecture that empowers domain experts to set up and customize collaborative information systems without programming based on layered business-oriented conceptual models. This architecture emerged from a decade of iterative research, design, development, and technology transfer projects which focused on individual modelling and system construction activities. This paper puts these results and publications into a larger architectural perspective, explains the rationale behind this architecture, and argues to consider it as a reference architecture for model-based collaborative information systems in general.*

Keywords. Reference Architecture • Model-Driven Engineering • Information Systems

## 1 Introduction

Since 2002, our team at the Technical University of Munich pursues the ambitious research goal to build enterprise-scale collaborative information systems based on business-oriented conceptual models only. This approach should empower business users to directly define and customize the systems they use for their day-to-day knowledge-intensive work in fields like engineering, product management, health-care, education, service management, or IT management without programming.

These models are the results of several cycles of development in collaboration with more than ten large and medium-sized enterprises in domains such as logistics and transportation, health care and insurance in Europe (see also Table 1).

This paper is structured as follows: In Section 2 we describe the notion (vision) of model-driven collaborative information systems from an end-user perspective and then explain in Section 3 how the implementation of such systems naturally leads to eight architectural layers which implement and encapsulate distinct capabilities: data storage, schema definition, access control definition, query capabilities, case modeling, case management, and user interface definition. We then elaborate the core abstractions to define and customize collaborative information systems using an integrated conceptual meta-model (Figure 3) which can be subdivided into models corresponding to the abstractions of the layers. In Section 5 we provide references to projects and scientific publications that report on lessons learned using the generic system in different application areas and sketch the use of the system as a platform for patient-centric health care. The paper ends with related work, a summary and an outlook on future work.

## 2 Collaborative Information Systems

We use the term collaborative information systems (CIS) to describe systems which allow different

* Corresponding author.
E-mail. adrian.hernandez@tum.de
Some of this work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement n° 689802).

stakeholders to collaboratively model and manage their information and knowledge-intensive processes. These processes can be internal to an organization (e. g., financial management, IT management) or can involve the cooperation of members of the organization with customers, suppliers and partner organizations (e. g., product design, customer relationship management, customer-centric healthcare).

In continuously changing environments, the understanding of the details of the information structures and work processes evolves over time, in particular in agile organizations and business ecosystems, which requires a consistency-preserving co-evolution of information, information structures, and information management processes. Consequently, collaborative information systems have to support both, structured and unstructured information (texts, hypertexts, diagrams, media files) as well as structured processes and semi-structured, adaptive case management.

Such a model-driven approach allows practitioners to create and reuse models of their problem domain within their running system instance. This results in the creation of an ecosystem specific to the problem and users can instantly collaborate inside the organization and cross-organization using a basic set of concepts (i. e., Workspaces, Wikis, and Pages). These models are directly deployed to a running system avoiding the hassles of model-to-code transformations and redeployments. Therefore, the enterprise can use domain experts without programming knowledge to quickly set-up a new information system. Furthermore, these systems also allow practitioners to dynamically update and evolve their models to meet the demands of the changing business needs. The model-driven architecture adopted by these systems allows to automatically update the persistence and presentation layers.

In our system implementation, we did not commit to a specific concrete (textual or graphical) syntax for configuring specific collaborative information systems. Instead of this, our generic system provides several (RESTful) APIs to create, read, update or delete (data, access, case, user-interface) models also at runtime. Most of our projects use interactive web-based front-ends where end-users can use form-based, text-based or graphical interfaces for model definition, execution and analysis (similar to Excel, Access, Sharepoint, and Camunda) which are then translated to API calls of our model-based back-end (as a service).

## 3 Layers and Capabilities of the Reference Architecture

Figure 1 provides an overview of the layers of the reference architecture. The graphical representation emphasizes that each layer adds new abstractions (e. g., an access control model) to the abstractions of the layers below. The abstractions are made accessible to clients via RESTful APIs for introspection or modification. Each concept (e. g., Principal, Group, User, Membership) is mapped to a separate resource in the REST API. The semantic relationships between the concepts (association, aggregation, sub-classing) are mapped to (bi-directional) links between resources or specialized resources.

The database symbols and the colored boxes within it indicate that the models are first-class entities in the persistent store. This leads to a nice orthogonality of concepts. For example, access control and version control also apply to higher-level concepts like case definitions.

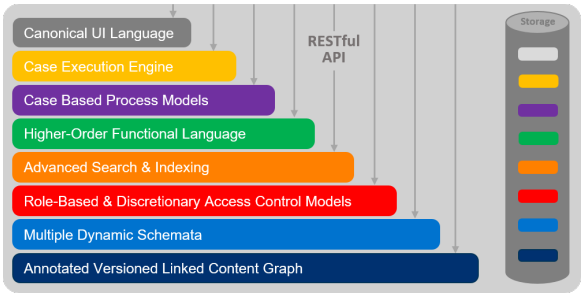



*Figure 1: Layers of the reference architecture.*

The same colour coding of the layers and the models is also used in Figure 3 in Section 4 to explain which semantic concept is realized at which layer of the architecture.

Three main principles heavily influenced our system design: Empowerment of domain experts and end-users, agility in the model/data and process evolution, and a secure self-contained environment for intra- and inter-organizational collaboration.

After several iterations within collaboration academic and industry projects, we derived a group of orthogonal concepts to ensure the flexibility and scalability of the system despite their expressiveness for modelling the enterprise data. The concepts are Workspaces, Pages, Entities, Attributes, Files, and Principals.

An instance of the system is used by an organization or a network of collaborating organizations. An arbitrary number of personal, team, project or organizational-unit workspaces can be created. This allows different communities of practice to model and manage their information and processes in their workspaces independently, or to selectively share information and models (e. g. CRM data and processes) across community boundaries.

The reference architecture comprises the following eight layers and capabilities:

**Annotated Versioned Linked Content Graph:** The first layer supports versioned data storage and ensures the capability of the system to store revisions semi-structured data in a form of entities (i. e., JSON objects) and references between them. The rationale is that companies already manage structured and unstructured data, so it is necessary to create a mechanism for the system to import an initial set of semi-structured data (e. g., Excel sheets, wiki pages with embedded tables and media) without strong data schema that can evolve over the time. This layer supports the data-first (schema second) approach for data modelling (Büchner 2007).

**Multiple Dynamic Schemata:** This layer allows the users of the information system to collaboratively structure their information over time by adding or removing schema constraints involving relationship and cardinality constraints as needed. A data schema is thus not seen as a definition of a container that is subsequently filled with data (like an SQL database), but as a collection of constraints imposed on a flexible content graph that evolves over time. However, this implies that at certain times during system evolution the data can contain inconsistencies, which can be resolved collaboratively without leaving the scope of the system. (Büchner 2007, Matthes et al. 2011)

**Role-Based & Discretionary Access Control:** This layer addresses the security concerns in the system. The security model comprises users and groups which can be internal or external to the organization. The access rights (administrator, editor, author, and reader roles) are defined initially at the Workspace level and can be overwritten (loosened or tightened) at the Entity level. The rationale is to guarantee the secure access to the data stored in the system.

**Advanced Search & Indexing:** In this layer, the unstructured data (i. e., long texts or hypertexts) is linked to the semi-structured data (using parsing and full-text indexing technologies). This is a core element of the Hybrid Wiki approach (Matthes et al. 2011) and lays the groundwork for natural language processing techniques and model discovery processes. The rationale behind this design is the fact that not all the data in the enterprise is structured or semi-structured and the system should be able to use all the possible formats of the data in the enterprise without the need of an upfront structuring process.

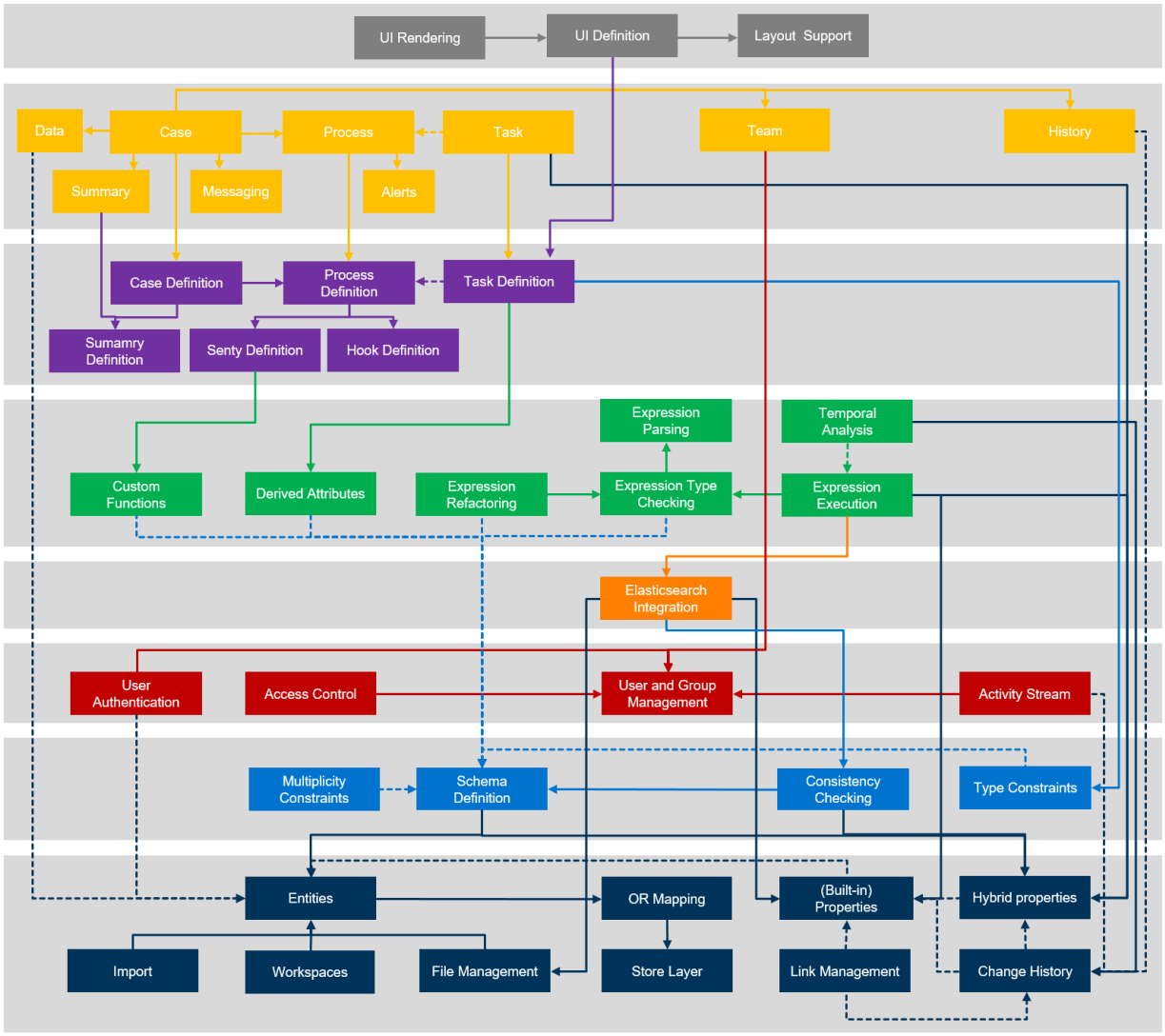



*Figure 2: Capabilities ordered according to conceptual layers. A solid line represents the usage of a certain capability and a dashed line represents extended functionality.*

**Higher-Order Functional Language:** This layer introduces a strongly-typed query language (similar to LINQ - Meijer et al. 2006) to access all the structured information in the information system. This language allows adding new attributes to Entities as a result of the computations (i. e., Derived Attributes) and using operations over collections of Entities such as Map, Reduce, and Join. The rationale behind that is creating a flexible mechanism to the user of the system to extract knowledge from the stored data, which is not limited by the basic API of the system, and allows the users to create flexible and tailored data views based on individual information demands. (Reschenhofer and Matthes 2016)

**Case Based Process Models:** This layer allows the user to define knowledge- and data-intensive processes following the adaptive case management paradigm (Swenson, Palmer et al. 2010). The user can define stages, human tasks and automated tasks that make up a collaborative

process. The tasks can link to one or multiple Attributes which are needed or modified by each task. Thereby, the system allows the user to define standard processes or reusable process fragments which can be dynamically instantiated as needed.

**Case Execution Engine:** This layer manages the information regarding the current states of all cases being executed in the system and the links to the shared or case-local data and the actors involved in a case. The case execution engine enforces the access control policies defined in the access rights layer and the data management layer keeps an audit trail of the executed steps and data modifications. The rationale is to keep track of the process steps execution and how the users accomplished the case also for future analysis (e. g. for compliance or prediction purposes).

**Canonical UI Language:** This layer allows the user to define links between the data and its representation using the predefined access control policies. This representation includes information about how the Attributes should be grouped and laid out visually. This ensures a clear separation between the data and its representation, thereby an Entity can have multiple representations based on the context of use. Therefore, Entities can be represented using vertical and horizontal layouts to display their attributes. The rationale is to capture the knowledge regarding the presentation of information in a model without the need to inspect specific UI implementations.

Figure 2 shows all the capabilities that are provided in each layer of the reference architecture and their semantic relationships (capability A uses capability B, capability A extends capability B). The colour coding links the elements to the layers depicted in Figure 1 and the concepts in the conceptual model in Figure 3. We found this capability map to be a very useful starting point for core developers of the platform to understand the key dependencies in the architecture.

## 4 Conceptual Meta-Model

The underlying meta-model of cooperative information systems (see Figure 3) builds on the already introduced core concepts of Workspaces as containers (name spaces) for Entities, EntityDefinitions, Attributes, and AttributeDefinitions. These concepts structure the model inside a Workspace and capture its current state. An Entity consists of a collection of Attributes, and the Attributes are stored as a key-value pair. The attributes have a name and can store an ordered list of values of different types, for example, strings or references to other Entities. The user can create an attribute at run-time to capture structured information about an Entity. An EntityDefinition allows users to refer to a collection of similar Entities and their common schema, e. g., organizations, persons, amongst others. The EntityDefinition consists of multiple AttributeDefinitions, which in turn contain multiple validators such as multiplicity validator, string value validator, and link value validator. Additionally, an individual Attribute and its values can be associated with validators for maintaining integrity constraints.

The EntityDefinition and AttributeDefinition are loosely coupled with Entity and Attribute respectively through their name. These elements specify soft-constraints on the Entities and Attributes. The use of soft-constraints implies that the users are not restrained by strict integrity constraints while capturing information in Entities and their Attributes. Therefore, the system can store a value violating integrity constraints as defined in the current model (e. g., during schema evolution stages, or after import of non-conformant data from an external data source).

A CaseDefinition is a template for all related case instances and links to a root EntityDefinition which describes the schema of the case data. Every CaseDefinition consists of multiple ProcessDefinitions that are either a complex StageDefinition, a single HumanTaskDefinition or an AutomatedTaskDefinition. StageDefinitions

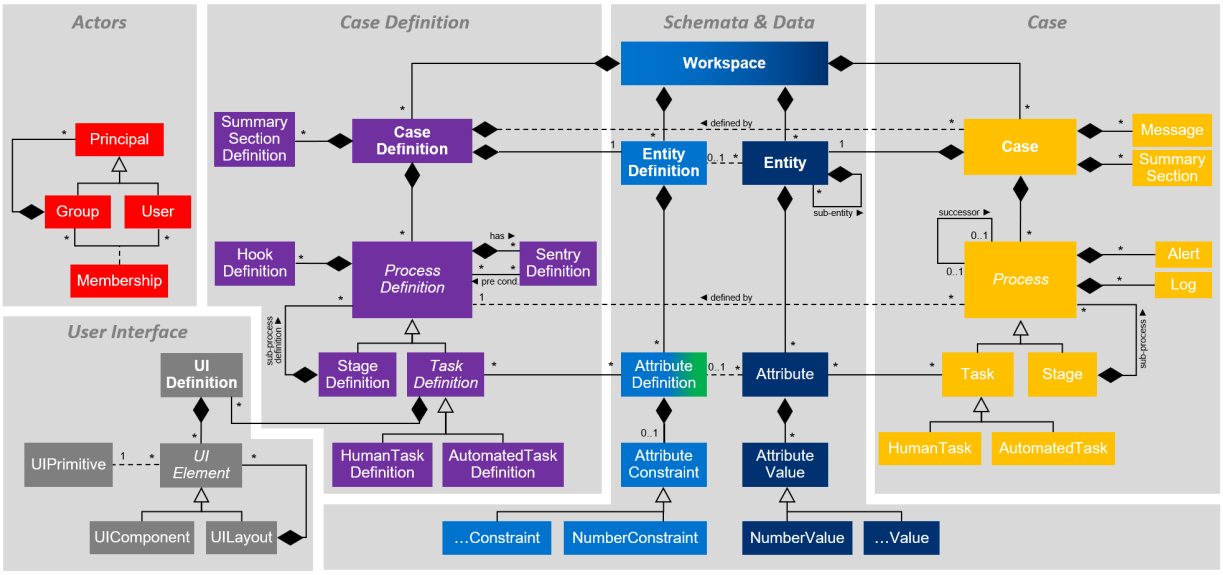



*Figure 3: Conceptual Meta-Model.*

represent a container that is used to group either sub StageDefinitions or TaskDefinitions. Preconditions to the activation of a stage are expressed as SentryDefinitions that either depend on some previously accomplished Process elements or expressions (Boolean predicates) related to the case data. In order to receive notifications on state changes of a Process element, it is possible to define HookDefinitions related to a Process-Definition. These hooks can be used to integrate services from third-party systems (with REST APIs). Additionally, SummarySectionDefintions are used to create short summaries (data views) based on the data created by the Case.

A Case is an instance of a CaseDefinition and follows the same structure. Each Case model element has a state, such as AVAILABLE, ENABLE, ACTIVE, COMPLETED or TERMINATED. Several model elements such as Message, Alert and Log exist only at the instance level. Cases are mostly knowledge-intensive and often lead to a discussion between all stakeholders involved, therefore Messages are attached to a case. The concept of Alerts helps to notify users involved in a Case, for example about overdue tasks. Also, third-party systems can create these alerts for guiding or alerting users. Runtime errors or exceptions are also stored persistently as Log elements.

A UIDefinition describes how attributes are to be presented to the user in a user interface (e. g., a web form). A UIDefinition is composed of a group a UIElements, every of which can be a single component or a layout element. A UI definition refers to an implementation of a software component that dynamically interprets UI model (i. e., a UI renderer component). Using this model, the knowledge regarding the presentation of the attributes is not tied to a specific technology. For example, the user can group attributes in a UILayout instance and provide configurations such as labels and validation errors that can be used at runtime by the UI renderer to create web forms using the Material Design[1] framework.

## 5 Applications of the reference architecture

In recent years, the reference architecture has been used in several business domains and in

[1] https://material.io/guidelines (Accessed on: 28/11/2017)

| | Annotated VersionedLinked Content Graph | Multiple Dynamic Schemata | Role-Based & Discretionary Access Control Models | Advanced Search & Indexing | Higher-Order Functional Language | Case Based Process Models | Case Execution Engine | Canonical UI Language | References |
|---|---|---|---|---|---|---|---|---|---|
| CONNECARE | ● | ● | ● | ● | ● | ● | ● | ◖ | Vargiu et al. 2017<br>Vargiu et al. 2018<br>http://www.connecare.eu |
| LEXALYZE | ● | ● | ● | ● | ● | ◖ | ◖ | ◖ | Waltl et al. 2017a, Waltl et al. 2017b<br>Waltl et al. 2016, Waltl et al. 2015<br>http://www.en.lexalyze.de |
| VolunteerApp | ● | ● | ● | ◐ | ● | ◖ | ◖ | ◖ | https://volunteers.in.tum.de |
| EcoSystem Explorer | ● | ● | ● | ● | ● | ◖ | ◖ | ● | Faber et al. 2017<br>Hernandez-Mendez et al. 2017<br>https://ecosystem-explorer.in.tum.de |
| Informatics Department intranet | ● | ● | ● | ● | ◖ | ◖ | ◖ | ◖ | https://intranet.in.tum.de |
| Chairs internal and external webpage | ● | ● | ● | ● | ◐ | ◖ | ◖ | ◖ | https://wwwmatthes.in.tum.de |
| Spreadsheet 2.0 | ● | ● | ● | ● | ● | ◖ | ◖ | ◖ | Reschenhofer et al. 2016b<br>Reschenhofer and Matthes 2016<br>Reschenhofer et al. 2016a<br>Reschenhofer 2017 |
| SyncPipes | ● | ● | ● | ● | ○ | ◖ | ◖ | ◖ | Bhat et al. 2016 |
| SmartNet Project | ● | ● | ● | ● | ◖ | ◖ | ◖ | ◖ | https://www.smart-nets.eu |
| Inteligent Contextual Mail | ● | ● | ● | ● | ● | ● | ● | ◖ | https://icm.in.tum.de |
| The Open EAM Knowledge Platform | ● | ● | ● | ● | ○ | ◖ | ◖ | ◖ | Bondel and Matthes 2017<br>http://www.eam-initiative.org |

Legend: ● used ◐ partly used ○ not used ◖ not available during developemend

*Table 1: Projects that use the reference architecture.*

different organizational contexts. Table 1 lists the most relevant use cases, indicates the layers of the architecture being used an provides references to scientific publications with more detailed information about the experience gained using the architecture in each use case.

In the remainder of this section, we explain the application of the reference architecture in the European health-care project CONNECARE which provides a digital platform for Personalized Connected Care for Complex Chronic Patients

that also allow gaining new medical insights based on the analysis of case studies. The general objective is to connect all professionals (doctors, nurses, etc.) directly with the patient via smartphone apps and wearable devices and manage the cases in a model-driven way via an adaptive case management platform. Therefore, three case study types are defined that are applied in four different hospitals that are spread across Europe. Every case study follows the same basic stages such as case identification, case evaluation, work plan definition, work plan execution and finally the discharge stage.

Figure 4 illustrates the workflow of one case study using the CMMN, the highlighted parts represent the basic stages of every case. Within every stage multiple tasks need to be accomplished by the clinicians, e. g. completing a Carlson[2] questionnaire that then automatically computes the related score. During the work plan stage, the clinicians can define tasks that need to be executed by the patient such as to do ten sit-ups per day. The patient performs these tasks assisted by a smartphone app and the clinician is notified about the execution results. For a comprehensive summary of the whole CONNECARE project consult (Vargiu et al. 2017, 2018).

Figure 5 shows the high-level CONNECARE architecture focusing on the Smart Adaptive Case Management system and its related systems. The main components are the Smart Adaptive Case Management (SACM) for all interactions with the professional, and the Self-Management System (SMS) for all interactions with the patient. The SACM system consists of several components such as: 1) *Professional Interface* provides information for professionals such as doctors, nurses etc. The Angular 2.0 client application of this interface uses the domain-specific API, 2) *Decision Support System* support clinicians during the treatment process, 3) *SocioCortex-Server* is an instance of the reference architecture described in this paper and is accessible by a JSON based RESTful API and is responsible for case modeling, case execution, access control, data storage and data modeling and for the integration of all other systems 4) *SACM-Backend* provides a deployment-specific functionality as well as an API for the Professional Interface and the Message Broker. The functionality of the SocioCortex-Server is wrapped and enhanced to provide deployment-specific functionality.

The SACM-Backend is communicating via a Message Broker with the central User Identity Management and the SMS. In the future, the information from the hospital information systems at the various deployment sites in Europe will be exchanged via the message broker as well. All components are deployed as Docker micro services on the Amazon EC2 cloud.

## 6 Related Work

To the best of our knowledge, this paper provides the first reference architecture for model-based collaborative information systems which integrates process and data modelling in a fully model-based system without depending on programming skills and suitable for productive use in industry.

However, this work builds on the vast knowledge that has been created from research in information systems, conceptual modelling and process modelling (Hesse and Mayr 2008; Reichert et al. 2007).

In the domain of information systems, we can highlight two compelling works that complement the concepts described in the proposed architecture. First, the concept of conceptual independence introduced by (McGinnes 2011) (McGinnes and Kapros 2015). This work describes the problems that arise when the internal software components and their corresponding conceptual models are highly coupled. Likewise,

[2] http://www.bgs.org.uk/pdfs/assessment/cci.pdf (Accessed on: 28/11/2017)

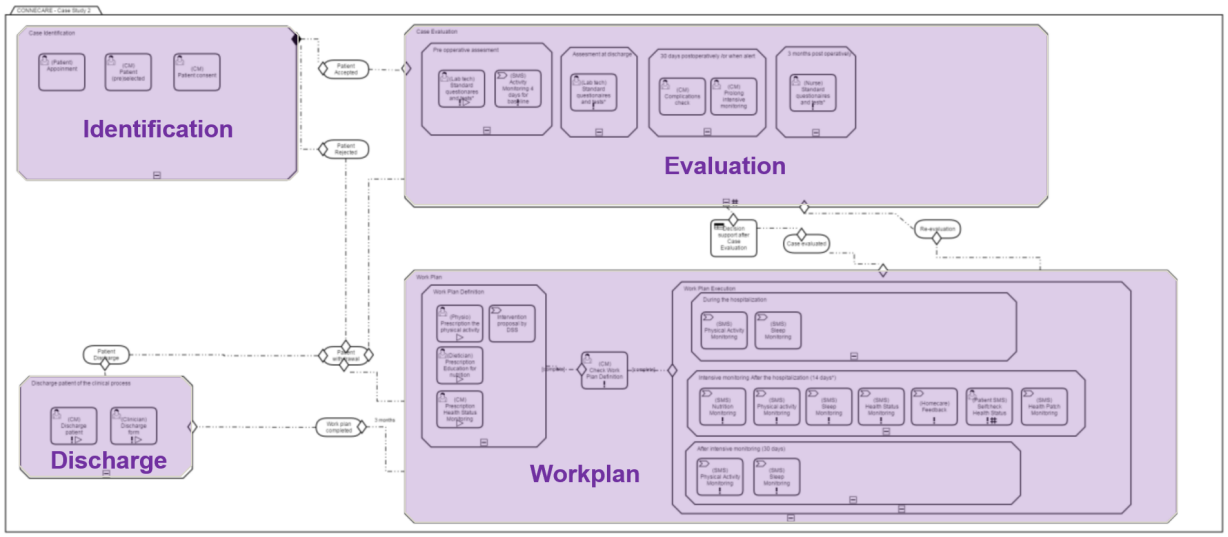



*Figure 4: CONNECARE case study sample modeled in CMMN notation.*

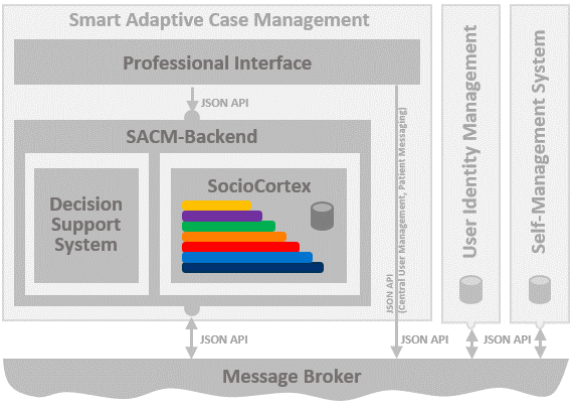



*Figure 5: CONNECARE Architecture with focus on the Smart Adaptive Case Management system.*

a type of adaptive information system is proposed to resolve the dependencies between the system and the conceptual models. This work shares the same problem that motivates our research. However, it is limited to the analysis of data models and not how the system manages the processes for the creation of knowledge. Second, the Social Knowledge Environments, introduced by (Pawlowski et al. 2014), establishes the benefits of the integration between social software and the creation of knowledge in the context of information systems and motivates the study of such integration. To our consideration, the proposed reference architecture contributes to the study that is motivated in (Pawlowski et al. 2014), because it was created from concrete examples of collaboration between academia and industry that allow analysing the different roles and social implications of the data modelling and knowledge creation processes.

From a process modelling perspective, it is worth to mention two approaches that could lay the ground for concrete implementations of the reference architecture. First, the Normalized Systems Theory, proposed by (Mannaert and Verelst 2009), which establishes a procedure for modelling systems considering evolvability principles. This approach can provide guidelines to experts how to create process models using the concrete implementation of the proposed reference architecture. The second prominent approach is Object-aware Business Processes proposed by (Künzle et al. 2010), and the PHILharmonic Flows framework (Carolina Ming Chiao et al. 2014). This approach provides building blocks to foster flexibility in process models, and has been applied in several contexts such as healthcare (C. M. Chiao et al. 2013b) and

university internal processes (C. M. Chiao et al. 2013a).

## 7 Conclusions

In this paper, we described the notion of model-driven collaborative information systems from an end-user perspective and explained how the implementation of such systems naturally leads to eight architectural layers which implement and encapsulate specific information modelling and management capabilities. We identified the core abstractions needed to define and customize collaborative information systems and developed an integrated conceptual meta-model which can be subdivided into models corresponding to the abstractions of the layers.

Based on the positive experience of a decade of practical (commercial) use of the architecture in several business domains and deployment contexts (intra-organizational, cross-organizational collaboration), we concluded that the architectures should be considered as a reference architecture for cooperative information systems.

In the future, we plan to integrate NLP, NLG and ML capabilities into the functional query layer and to study how deontic models and so-called smart contract languages fit into the layered architecture of model-based collaborative information systems.

## References

Bhat M., Shumaiev K., Biesdorf A., Hohenstein U., Hassel M., Matthes F. (2016) Meta-model based framework for architectural knowledge management. In: Proccedings of the 10th European Conference on Software Architecture Workshops. ACM, p. 12

Bondel G., Matthes F. (2017) Hybride Wikis als Repository für IT-Unternehmensarchitektur. In: IT-Unternehmensarchitektur: Von der Geschäftsstrategie zur optimalen IT-Unterstützung

Büchner T. (2007) Introspektive modellgetriebene Softwareentwicklung. Dissertation, Technische Universität München, München

Chiao C. M., Künzle V., Reichert M. (2013a) Integrated modeling of process- and data-centric software systems with PHILharmonicFlows. In: 2013 IEEE 1st International Workshop on Communicating Business Process and Software Models: Quality, Understandability, and Maintainability, CPSM 2013. IEEE Computer Society, Eindhoven

Chiao C. M., Künzle V., Reichert M., Künzle V. (2013b) Object-aware process support in healthcare information systems: Requirements, conceptual framework and examples. In: International Journal on Advances in Life Sciences 5(1-2), pp. 11–26

Chiao C. M., Küenzle V., Reichert M. (2014) Towards schema evolution in object-aware process management systems. In: Enterprise modelling and information systems architectures - EMISA 2014, Luxembourg, September 25-26, 2014, pp. 101–115

Faber A., Hernandez-Mendez A., Matthes F. (2017) Towards an Understanding of the Connected Mobility Ecosystem from a German Perspective. In: ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems, Volume 3, Porto, Portugal, April 26-29, 2017, pp. 543–549

Hernandez-Mendez A., Faber A., Matthes F. (2017) Towards a Data Science Environment for Modeling Business Ecosystems: The Connected Mobility Case. In: Advances in Databases and Information Systems. Springer, pp. 324–330

Hesse W., Mayr H. C. (2008) Modellierung in der Softwaretechnik: eine Bestandsaufnahme. In: Informatik-Spektrum 31(5), pp. 377–393

Künzle V., Weber B., Reichert M. (2010) Object-aware Business Processes: Properties, Requirements, Existing Approaches. Technical Report UIB-2010-06. University of Ulm. Ulm, Germany

Mannaert H., Verelst J. (2009) Normalized Systems: Re-creating Information Technology Based on Laws for Software Evolvability. Koppa

Matthes F., Neubert C., Steinhoff A. (2011) Hybrid Wikis: Empowering Users to Collaboratively Structure Information.. In: ICSOFT (1) 11, pp. 250–259

McGinnes S. (2011) Conceptual modelling for web information systems: What semantics can be shared? In: De Troyer O., Bauzer Medeiros C., Billen R., Hallot P., Simitsis A., Van Mingroot H. (eds.) 30th International Conference on Conceptual Modeling. Lecture Notes in Computer Science Vol. 6999. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 4–13

McGinnes S., Kapros E. (2015) Conceptual independence: A design principle for the construction of adaptive information systems. In: Information Systems 47, pp. 33–50

Meijer E., Beckman B., Bierman G. (2006) LINQ: Reconciling Object, Relations and XML in the .NET Framework. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data. SIGMOD '06. ACM, Chicago, IL, USA, pp. 706–706

Pawlowski J. M., Bick M., Peinl R., Thalmann S., Maier R., Hetmank L., Kruse P., Martensen M., Pirkkalainen H. (2014) Social Knowledge Environments. In: Business & Information Systems Engineering 6(2), pp. 81–88

Reichert M., Strecker S., Turowski K. (2007) Proceedings of the 2nd Int'l Workshop on Enterprise Modelling and Information Systems Architectures - Concepts and Applications (EMISA'07). Lecture Notes in Informatics LNCS4549. GI - Gesellschaft fur Informatik

Reschenhofer T. (2017) Empowering End-users to Collaboratively Analyze Evolving Complex Linked Data. PhD thesis, Technische Universität München

Reschenhofer T., Bürgin P., Matthes F. (2016a) A Social Information Flow Graph: Design and Prototypical Implementation.. In: CAiSE Forum, pp. 137–144

Reschenhofer T., Matthes F. (2016) Supporting end-users in defining complex queries on evolving and domain-specific data models. In: 2016 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2016, Cambridge, United Kingdom, September 4-8, 2016, pp. 96–100

Reschenhofer T., Waltl B., Gjorgievska S., Matthes F. (2016b) A Semantic Meta Model of Spreadsheets.. In: ECIS, ResearchPaper90

Swenson K. D., Palmer N. et al. (2010) Mastering the unpredictable: how adaptive case management will revolutionize the way that knowledge workers get things done Vol. 1. Meghan-Kiffer Press Tampa

Vargiu E., Fernández J., Miralles F., Cano I., Gimeno-Santos E., Hernandez C., Torres G., J. Colomina J. d. B., Kaye R., Azaria B., Nakar S., Lahr M., Metting E., Jager M., Meetsma H., Mariani S., Mamei M., Zambonelli F., Michel F., Matthes F., Goulden J., Eaglesham J., Lowe C. (2017) Integrated care for complex chronic patients background. In: ICIC17 – 17th International Conference on Integrated Care, Dublin. International Foundation for Integrated Care

Vargiu E., Fernández J., Miralles F., Haim R., Nakar S., Weijers V., Meetsma H., Mariani S., Mamei M., Zambonelli F., Michel F., Kelly J., Eaglesham J. (2018) Patient Empowerment and Case Management in CONNECARE. In: GCIC18 – Global Conference on Integrated Care, 2018, Singapore

Waltl B., Landthaler J., Matthes F. (2016) Differentiation and Empirical Analysis of Reference Types in Legal Documents. In: Legal Knowledge and Information Systems - JURIX 2016: The Twenty-Ninth Annual Conference, pp. 211–214

Waltl B., Reschenhofer T., Matthes F. (2017a) Process and Tool-Support to Collaboratively Formalize Statutory Texts by Executable Models. In: International Conference on Database and Expert Systems Applications. Springer, pp. 118–125

Waltl B., Reschenhofer T., Matthes F. (2017b) Process and Tool-Support to Collaboratively Formalize Statutory Texts by Executable Models. In: Database and Expert Systems Applications - 28th International Conference, DEXA 2017, Lyon, France, August 28-31, 2017, Proceedings, Part II, pp. 118–125

Waltl B., Zec M., Matthes F. (2015) A Data Science Environment for Legal Texts. In: Legal Knowledge and Information Systems - JURIX 2015: The Twenty-Eighth Annual Conference, Braga, Portual, December 10-11, 2015, pp. 193–194

# From Requirements to Code: A Conceptual Model-based Approach for Automating the Software Production Process

Oscar Pastor[*,a], Marcela Ruiz[b]

[a] PROS Research Centre, Universitat Politecnica de Valencia, Spain
[b] Utrecht University, the Netherlands

Abstract. *Conceptual Models are part of an increasing number of engineering processes. The model driven development approach considers conceptual models as first-class entities and also considers tools, repositories, etc. as models. In order to take full advantage of these ideas, model transformation is a main activity. A sound software production process, conceptual-modelling based, must go from the initial requirements model to the final application code through a well-defined set of conceptual models and transformations between them. Model transformation aims at supporting the production of target models from a number of source models, while keeping a full traceability support. The current paper presents a practical application of these ideas using the Model Centred Architecture contributed by Heinrich C. Mayr. In this line, we present our research efforts on the integration of requirements and executable conceptual models. We reflect on the integration of Communication Analysis (a communication-oriented business process modelling and requirements method) and the OO-Method (an object-oriented model-driven development method).*

Keywords. Conceptual Modelling • Conceptual Programming • OO-Method • Communication Analysis • Model-Centred Architecture • Model-Based Software Production

## 1 Introduction

An essential component of a conceptual model-based software production approach is an executable conceptual model. This is how we start this paper, presenting the OO-Method approach in section 2. To avoid moving from an Extreme Programming perspective to a kind of Extreme Conceptual Modelling point of view, where a particular conceptual model comes from must be clearly determined. This is why we need a concrete requirements modelling strategy (presented through the Communication Analysis perspective in section 3), and a concrete model transformation from the requirements model to its corresponding executable conceptual schema. This is what we discuss in section 4 using a model centred architecture. Concluding remarks and references complete the work.

## 2 The OO-Method approach

OO-Method is an approach for automatic software generation based on the specification of object-oriented conceptual models (Pastor and Molina 2007). It is supported by Integranova [1] , a model-driven tool that provides an OO-Method modelling environment, a conceptual model compiler, and automatic code generation. OO-Method uses four conceptual models partial views that conform all together a complete information systems specification: i) Object model, ii) Dynamic model, iii) Functional model, and iv) Presentation model.

* Corresponding author.
E-mail. opastor@pros.upv.es

[1] http://www.integranova.com/

The conceptual models are platform independent, i.e., they do not involve platform-dependent characteristics.

**Object model**

This model specifies the structure and static relationships between the classes of a software system. It provides a graphical notation that can be considered equivalent to a UML class diagram where only a delimited set of primitive constructs are selected as relevant. It basically includes classes, attributes, services (events (simple) and transactions or operations (complex)), and the relationships between the classes.

The basic graphical notation is complemented with the specification of textual integrity constraints written according to a well-defined first-order logical language. As an example, a rent-a-car system will include in its object model classes as car, customer, etc., each one with its corresponding set of attributes and services.

**Dynamic model**

This model specifies the dynamic and behavioural aspects of the classes of the object model. The dynamic model can be considered equivalent to UML's state transitions diagram. The valid life-cycles of the classes are represented in this model, as well as the possible interactions between different objects (i. e., instances of different classes).

**Functional model**

This model specifies the semantics of the change of an object state as a result of event executions. For example, a change in the state of an object car (from available to rented) when the rent service is activated. A set of services, preconditions, and post-conditions are specified to define changes in object states. OO-Method provides a declarative language to indicate the constraints that are related to each object state.

The functional model provides the facilities to specify domain dependent restrictions. The declarative language of the functional model is aligned with the object and functional models.

**Presentation model**

This model specifies the characteristics of the user interface of a software system and how the users will interact with the system; the model is created by means of a pattern-based graphical model through three levels of detail, from more general to more specific characteristics.

Applications generated from conceptual modelling with the OO-Method follows a three-layer software architecture. The presentation layer contains the software components responsible for presenting users the application interface to interact with the software system. The application layer provides services that implement the functionality. The persistence layer provides the services that manage data persistence, in order to store and obtain the pieces of data necessary for the execution of an application.

The software process of the OO-Method consists of two stages. First, system analysts (modellers) create a conceptual schema, which corresponds to a representation of the problem space (i.e., the application domain). A UML-based notation and textual specifications are used according with the four model views commented before. Second, the code of an application is generated on the basis of the Execution Model of Integranova, which corresponds to a representation of the solution space and can be targeted at different technologies.

When comparing the OO-Method with other approaches for software modelling and development, it deals with the static (data-oriented), the dynamic and functional (behaviour-oriented) and the presentation views of an Information System (IS). All together they constitute a complete IS modelling and development approach. In addition, it relies on an underlying formal model OASIS; (Lopez et al. 1992) and it provides a conceptual model compiler intended to make true the conceptual programming goal that states that "the conceptual model is the code" (instead of "the code is the model") (Embley et al. 2011). This strategy allows the generation of complete and

ready-for-running applications by precisely specifying an IS through its conceptual model.

In relation to MDA (Model Driven Architecture; (Miller and Mukerji 2003)), a detailed description of its correspondence with OO-Method can be found in (Pastor and Molina 2007). The main points are that: 1) an OO-Method conceptual schema corresponds to a Platform-Independent-Model; 2) the execution model corresponds to a Platform-Specific-Model, and; 3) the code generated corresponds to an Implementation Model.

## 3 The Requirements Perspective: Communication Analysis

This section presents the Communication Analysis (CA) method (España et al. 2009). The CA-based approach extends the platform-independent view provided by the OO-Method with the upper Computation-independent model, where requirements modelling is going to be managed. To better understand the underlying ideas from a practical point of view, we are going to use a lab demo to illustrate the use of the Communication Analysis method.

The case presented in this paper is an adaptation of The SuperStationery Co. case. SuperStationery Co. is a company that provides stationery and office material to its clients. The company acts as an intermediary: the company has a catalogue of products that are bought from suppliers and sold to clients. This case is presented in full detail in (España et al. 2011). In this paper, we focus on the part of the sales manager business process (acronym SALE).

### 3.1 Concepts of Communication Analysis requirements models

To facilitate understanding of the illustrative example, this subsection presents a brief explanation of the concepts used for Communication Analysis (Communicative Event Diagrams and Message Structures).

**Concepts of Communicative Event Diagrams**

The Communicative Event Diagram (CED) is a business process modelling technique that adopts a communicational perspective and facilitates the development of an IS that will support those business processes (España 2011) (González et al. 2009). A communicative event is a set of actions that are related to information (acquisition, storage, processing, retrieval and/or distribution), which are carried out in a complete and uninterrupted way.

The unity criteria allows communicative events to be identified. Each communicative event is represented as a rounded rectangle and is given an identifier, a number and a descriptive name (e.g. SALE 1 in Figure 1).

For each event, the actors involved are identified. The primary actor triggers the communicative event and provides the input information. For instance, the client is the primary actor of the communicative event SALE 1.

The interface actor is in charge of physically interacting with the IS interface. Interface actors are specified at the bottom of the event. For instance, the salesman is the interface actor of the communicative event SALE 1. The receiver actors are those who need to be informed of the occurrence of an event. The sales manager is the receiver actor of the communicative event SALE 1.

An ingoing relationship is a communicative interaction that feeds the IS memory with new meaningful information. The main direction of the ingoing communicative interaction is from the primary actor to its related communicative event. For instance, the relationship named ORDER between the primary actor client and the communicative event SALE 1 is an ingoing communicative interaction. An outgoing relationship is a communicative interaction that consults the IS memory.

The main direction of the outgoing communicative interaction is from the communicative event to its related receiver actor. For instance, the relationship named ORDER that is between the communicative event SALE 1 and the receiving

actor SALES MANAGER is an outgoing communicative interaction.

The precedence relationships are represented as arrows among communicative events (e.g. SALE 1 requires the previous occurrence of PROD 2 and CLIE 1).

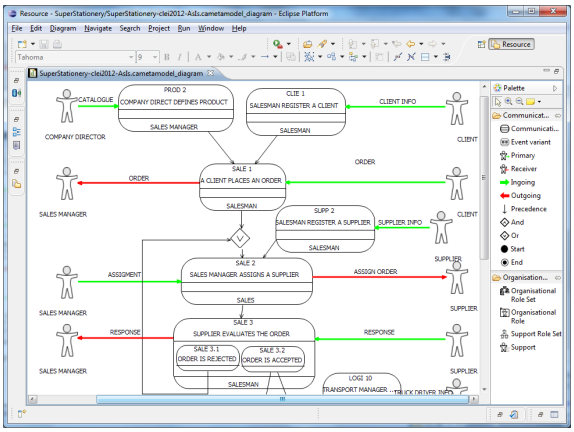


*Figure 1: CED of the Sale process of SuperStationery Co.*

### Concepts of Message Structures

Message Structures is a specification technique that allows the message that is associated to a communicative interaction to be described (González et al. 2011). A substructure is an element that is part of a message structure. This way, LINE, Client and Payment type are substructures that are part of the substructure ORDER (See Figure 2).

There are two classes of substructures: fields and complex substructures. A field is a basic informational element of the message that is not composed of other elements. A data field is a field that represents a piece of data with a basic domain. For instance, payment type is a data field. A reference field is a field whose domain is a type of business object. For instance, Client refers to a client that is already known by the IS.

A complex substructure is any substructure that has an internal composition. An aggregation substructure specifies a composition of several substructures. It is represented by angle brackets < >. For instance, LINE=<Product+Price+Quantity> specifies that an order line consists of information about a product, its price, and the quantity. An iteration substructure specifies a set of repetition of the substructures it contains. It is represented by curly brackets { }. For instance, an ORDER can have several lines for each product requested.

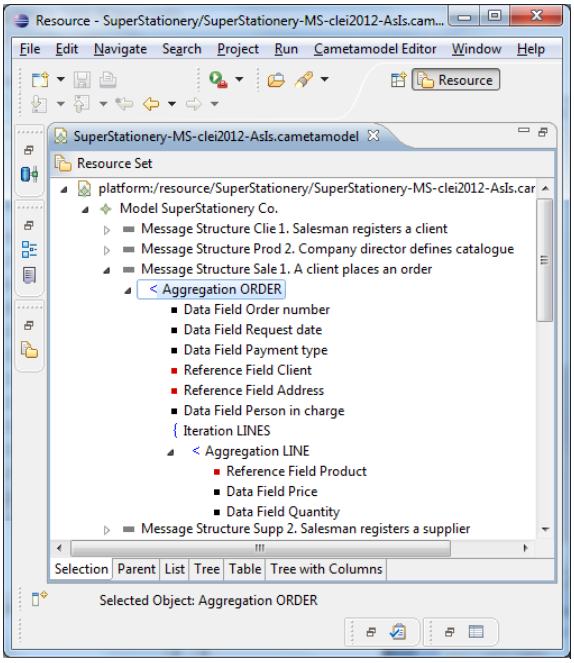



*Figure 2: Message Structure of the communicative event Sale 1*

The Communication Analysis Method is supported by means of the GREAT Process Modeller (Rueda et al. 2015). GREAT allows creating communicative event diagrams (i.e. business process models), specifying message structures (which describe the messages associated to each communicative event), and automatically generating a class diagram (representing the data model of an information system that would support organisational communication).

## 4 Generating an Executable Conceptual Model

España (2011) integrates the Communication Analysis and the OO-Method. The integration strategy is based on an ontological alignment of the concept

definitions in which the two methods are grounded. As a result, the ontological alignment provides precise transformation guidelines to transform Communication Analysis models into OO-Method object models (see Figure 3). The systematic derivation of OO-Method conceptual models from Communication Analysis requirements models is offered in two flavours: a set of rules to be manually applied by a human analyst, and an ATL model transformation that automates this task (Jouault and Kurtev 2006).

This is where the link between requirement modelling and conceptual model execution is established in a precise way through the connection between a CA model and its associated OOM model, the result of the transformation.

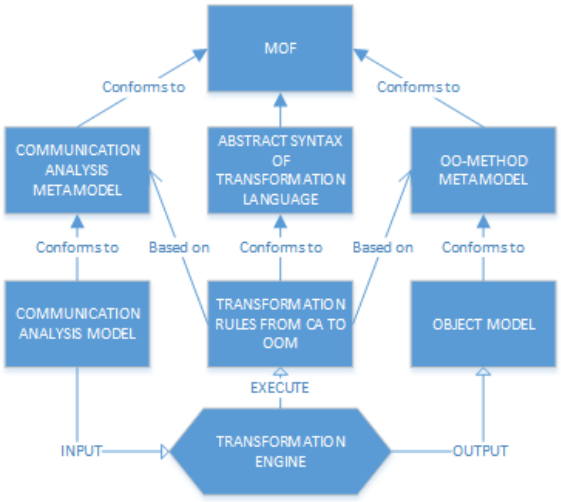



*Figure 3: Model derivation strategy from (Jouault and Kurtev 2006)*

### 4.1 Model Centred Architecture of Communication Analysis and OO-Method

Our research line applies the core concepts of the Model Centred Architecture paradigm (Mayr et al. 2017). As a result, the requirement and conceptual models of Communication Analysis and OO-Method can be seen as the core of information systems. Figure 4 presents an overview of the Model Centred Architecture paradigm applied to the Integration of Communication Analysis and OO-Method.

The Model Centred Architecture for the integration of Communication Analysis and OO-Method ensures a complete support for domain-specific applications. Thanks to model-driven tools like GREAT and Integranova, it is possible to give the power to system users and metamodel authors to specify and maintain information systems.

## 5 Conclusions

The current paper reflects on our research efforts for automating the software production process. The practical application of the Model Centred Architecture is reflected in the integration of the model-driven methods Communication Analysis and OO-Method. The application of the Model Centred Architecture benefits the role of system users by facilitating the maintenance and usage of model-driven tools.

In the near future we plan to explore the application of round trip model transformations. The idea is to involve system users when applying the transformation engines from requirements to code. The involvement of end users in the transformation process will ensure the implementation of domain specific requirements that should be specified during the transformation process. In addition, we plan to integrate a tool support for goal modelling to our current software production process. With the intentional perspective, the entire Model Centred Architecture acquires different views that will enrich the final software product.

We plan to improve the metamodel and model exchange interfaces support. The idea is to facilitate the connection among the different requirement specifications (processes, goals, and conceptual models) and software code. In this line, one big challenge will be the design of a modelling environment in which the traceability links become explicit assets.

In the long term, we intend to implement an evolution support for metamodels, models, and domain models. In this future scenario, the system

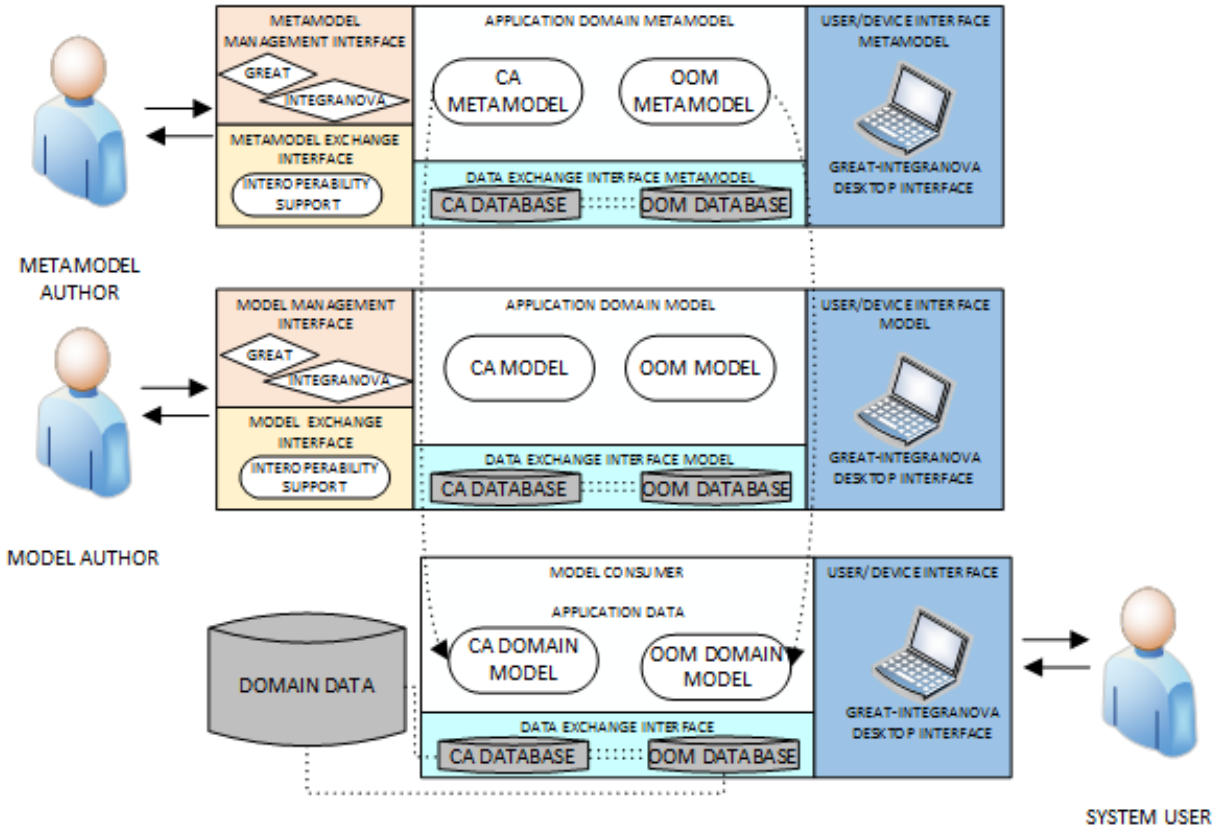



*Figure 4: Model Centred Architecture for Communication Analysis and OO-Method*

users will be in charge of the maintenance of the three layers of the Model Centred Architecture.

## References

Embley D. W., Liddle S. W., Pastor O. (2011) Conceptual-Model Programming: A Manifesto In: Handbook of Conceptual Modeling: Theory, Practice, and Research Challenges Embley D. W., Thalheim B. (eds.) Springer, pp. 3–16 https://doi.org/10.1007/978-3-642-15865-0_1

España S. (2011) Methodological Integration of Communication Analysis into a Model-Driven Software Development Framework. PhD thesis, Universitat Politecnica de Valencia http://hdl.handle.net/10251/14572

España S., González A., Pastor O. (2009) Communication Analysis: A Requirements Engineering Method for Information Systems In: Advanced Information Systems Engineering: 21st International Conference, CAiSE 2009 van Eck P., Gordijn J., Wieringa R. (eds.) Springer Berlin Heidelberg, pp. 530–545

España S., González A., Pastor O., Ruiz M. (2011) Integration of Communication Analysis and the OO Method: Manual derivation of the Conceptual Model. The SuperStationery Co. lab demo. In: CoRR abs/1101.0105 http://arxiv.org/abs/1101.0105

González A., España S., Pastor O. (2009) Unity criteria for Business Process Modelling: a theoretical argumentation for a Software Engineering recurrent problem. In: IEEE, p. 10

González A., Ruiz M., España S., Pastor O. (2011) Message Structures a modelling technique for information systems analysis and design. In: p. 12

Jouault F., Kurtev I. (2006) Transforming Models with ATL. In: Bruel J.-M. (ed.) Satellite Events at the MoDELS 2005 Conference. Springer Berlin Heidelberg, pp. 128–138 https://doi.org/10.1007/11663430_14

Lopez O. P., Hayes F., Bear S. (1992) Oasis: An object-oriented specification language. In: Loucopoulos P. (ed.) Advanced Information Systems Engineering. Springer Berlin Heidelberg, pp. 348–363 https://doi.org/10.1007/BFb0035141

Mayr H. C., Michael J., Ranasinghe S., Shekhovtsov V. A., Steinberger C. (2017) Model Centered Architecture In: Conceptual Modeling Perspectives Cabot J., Gómez C., Pastor O., Sancho M. R., Teniente E. (eds.) Springer International Publishing, pp. 85–104 https://doi.org/10.1007/978-3-319-67271-7_7

Miller J., Mukerji J. (2003) MDA Guide Version 1.0.1.. Object Management Group (OMG)

Pastor O., Molina J. C. (2007) Model-Driven Architecture in Practice. Springer-Verlag Berlin Heidelberg, Upper Saddle River, NJ, p. 302 https://doi.org/10.1007/978-3-540-71868-0

Rueda U., España S., Ruiz M. (2015) GREAT Process Modeller user manual. In: ArXiv e-prints http://adsabs.harvard.edu/abs/2015arXiv150207693R

# Roundtrip engineering of NoSQL databases

Jacky Akoka*,[a], Isabelle Comyn-Wattiau[b]

[a] CEDRIC-CNAM & TEM-Institut Mines Telecom, Paris, France
[b] ESSEC Business School, Cergy-Pontoise, France

Abstract. *In this article we present a framework describing a roundtrip engineering process for NoSQL database systems. This framework, based on the Model Driven Engineering approach, is composed of a knowledge base guiding the roundtrip process. Starting from a roundtrip generic scenario, we propose several roundtrip scenarios combining forward and reverse engineering processes. We illustrate our approach with an example related to a property graph database. The illustrative scenario consists of successive steps of model enrichment combined with forward and reverse engineering processes. Future research will consist in designing and implementing the main components of the knowledge base.*

Keywords. Roundtrip engineering • forward engineering • reverse engineering • roundtrip process • knowledge base • NoSQL database

## 1 Introduction

As stated by (Mayr et al. 2017), "models are the fundamental human instruments for managing complexity and understanding" . Model driven engineering (MDE) is considered as a methodology providing several benefits such as an improved code quality and a better traceability. It consists of the application of models to increase the level of abstraction required to develop and evolve software products. Its aim is to offer software development approaches in which abstract models of software systems are created and transformed facilitating their implementations. MDE is based on model transformation which takes one or more source models and transform them into one or more target models.

Roundtrip engineering (RTE) represents one facet of MDE. Since code and model are interrelated, changing code will change the model and vice versa. RTE can be considered as a way to improve the software engineering process. It consists mainly of forward engineering and reverse engineering. Forward engineering transforms conceptual models into source code. With reverse engineering, the source code is transformed back into conceptual models. The combination of the two paths leads to roundtrip engineering, keeping the two views consistent (Booch et al. 1998). Demeyer et al. (1999) defines RTE as the seamless integration between design diagrams and source code, between modeling and implementation. Therefore the aim of RTE is to enable a homogeneous integration between the design and the implementation phases. Code generation, described as a ***push method***, is obtained using forward engineering. The transformation of the source code into a conceptual model is obtained by a reverse engineering process based on a ***pull method***. Round-trip engineering corresponds to a ***push-pull method***.

RTE has been first used with UML. It has been extended to other technologies such as graphical user interface design, database design, and to other software modeling artifacts. RTE is different from the addition of forward and reverse engineering. Optimizing forward and reverse engineering leads to incremental transformation. Only the changed modules are transformed, rather than all artifacts.

* Corresponding author.
E-mail. jacky.akoka@lecnam.net

The RTE process preserves the information in the target artifact on the return trip. RTE tends not only to transform models but also to reconcile them. It automatically maintains the consistency of changing software artifacts. In other words, changes made to one model are propagated and reflected in another model using model transformation technologies (Sendall and Kozaczynski 2003; Czarnecki and Helsen 2003). RTE is a solution enabling the synchronization of models by keeping them consistent, thus maintaining conceptual-implementation mappings under evolution. RTE focuses mainly on synchronization. According to (Sendall and Kozaczynski 2003), RTE can be divided into three steps:

- Deciding whether a model under consideration has been changed,
- Deciding whether the changes cause any inconsistencies with the other models, and
- Once inconsistencies have been detected, updating the other models so that they become consistent again.

RTE is supported by methodologies and tools. However most development tools only offer very limited support. This is most probably a consequence of the difficulty in keeping multiple changing artifacts consistent.

The main advantage of RTE is that the design and implementation artifacts are automatically synchronized all the time (Kellokoski 2000). RTE promotes design-led development and improves design traceability since it enables an automatic generation of source code from conceptual models and automatic generation of the latter from source code. One major advantage is a short time to market and an improved quality of the software product. RTE improves the software process as well as its automation. Several qualities are expected from RTE approaches such as the ability to manage trace information and to facilitate the detection of conflicts between RTE activities.

So far, there has been very little research on roundtrip engineering of NoSQL database design. The aim of this paper is to start filling this gap. In particular, we propose a framework facilitating the roundtrip process and we derive requirements that this framework implies. Thus, it can be seen as a roadmap for a roundtrip engineering process of NoSQL databases. This article is organized as follows: The following section presents the state of the art related to RTE. Our framework is described in Section 3. We then show the results of its application using an illustrative scenario in Section 4. We derive our framework in Section 5. We finally indicate in Section 6 some conclusions and future work.

## 2 Related work

As we pointed out in the introduction, the basis of RTE is a clear definition of required consistency between the models. Therefore issues in RTE are closely related to those of consistency management. The latter is a technique for ensuring that models are consistent (Sendall and Kozaczynski 2003). The methodology presented in (Engels et al. 2001; Küster 2004) can be applied to define consistency for a given set of UML models. (Aßmann 2003) introduces mathematical definitions for RTE. The Fujaba System (Nickel et al. 2000) supports RTE for class diagrams. The CODEX system (Larrson and Burbeck 2003) aims at keeping a model consistent with a set of views on the model. The authors propose a RTE which essentially synchronizes models by keeping them consistent for maintaining conceptual-relational mappings. The added value of (Ciccozzi et al. 2011) is to ensure that extra-functional concerns modeled at design level are preserved at code execution level. They introduce a back annotation model containing information related both to traceability and monitoring results. (Bork et al. 2008) describe an approach towards model and source code RTE. The approach is based on reverse engineering of model-to-transformation (M2T) templates. They use (customizable) code generation templates as a grammar to parse the generated (and later modified) code. (Greiner et al. 2016) propose an RTE approach requiring the specification of QVT-R rules that relate two elements of

the respective meta-models. (Angyal et al. 2008) present an approach for model and code RTE based on differencing and merging abstract syntax trees (AST). (Antkiewicz and Czarnecki 2006) propose an RTE approach based on framework-specific modeling languages. (Hettel et al. 2009) propose an approach towards model RTE based on abductive logic programming. (Macedo and Cunha 2016) proposed an idea on how to circumvent some problems that are related to a QVT-R script by using the language Alloy with their tool Echo. In (Buchmann and Westfechtel 2013), the authors examine a standard use case: incremental round-trip engineering between design models and source code. More specifically, they address the coupling between a UML class diagram and a Java source code. Both model and code may be edited concurrently, and changes may be propagated back and forth to maintain consistency.

Although there are a number of round trip engineering tools available, only a few of them have been adopted by the developers' community. (Nagowah et al. 2013) present a state of the art of these tools, including (Borland Software Solutions 2009; Rational Software 2006; ArgoUML (Odutola et al. 2001); Gentleware AG Poseidon For UML (Boger et al. 2007); JBoss Seam (Orshalick and Assar 2010); Spring Web MVC framework (Winterfeldt 2012); AndroMDA 2012).

RTE is only one aspect of MDE. A general definition of the latter is given by (Hailpern and Tarr 2006). A characterization of MDE can be found in (Ruiz et al. 2017). In the context of MDE, model transformation plays an essential role. It defines transformations rules between a source and a target metamodel (Czarnecki and Helsen 2006). Model transformation includes code generation (Kleppe et al. 2003), models synchronization, in particular at the same or at different levels of abstraction (Ivkovic and Kontogiannis 2004), model evolution (Zhang et al. 2005) and reverse engineering from physical and/or logical levels to conceptual levels and vice versa (Favre 2004).

As it can be seen from this literature review, to the best of our knowledge, there is no approach related to RTE of NoSQL databases. This is precisely the aim of our approach described below.

## 3 Toward roundtrip engineering of NoSQL databases

It is generally admitted that NoSQL databases offer a flexible and a scalable solution to store and query structured, semi-structured and unstructured data. These databases offer a high level of query performance. They can be designed to meet the requirements of BI applications and analytics. Since they become more and more mature, they have been adopted by many companies. Maintaining these databases require methodologies offering some consistencies between their design models and their implementation. MDE represents such a methodology. It describes a system under consideration by means of high-level models. The latter are refined into low-level models until their level of detail is conform to the underlying platform. A model transformation is the process of mapping one input model into an output model. This transformation process requires the specification of transformation rules referring to metamodels. Therefore, a model transformation defines a set of rules to be applied between source and target metamodels. Transformations can be either unidirectional or bidirectional. Unidirectional transformations allow to map source metamodels to target metamodels, but not the other way around. Bidirectional transformations define mappings in both directions. This bidirectional transformation between meta-models is the main principle underlying roundtrip engineering. The goal of the latter is to keep a set of related metamodels synchronized. Whenever a change is applied to one metamodel, the other metamodels need to be adjusted to restore a consistent state.

For a specific application, roundtrip engineering is a method which allows the automatic synchronization of the source code after modification of the conceptual and/or the logical model and vice versa. Roundtrip engineering allows a bi-transformation between the model and the source code. Besides, roundtrip engineering is a way to

optimize the corrective, adaptive, evolutionary or perfective maintenance of applications. Indeed, anomalies can be detected, functionalities added and/or removed, new tests performed, and migrations to new platforms required during the life cycle of the application.

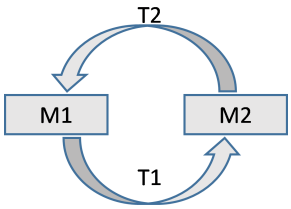



*Figure 1: Roundtrip generic scenario*

Our roundtrip framework is composed of different transformation rule sets which are all based on a common generic scenario (Figure 1) in which two metamodels (M1 and M2) are connected by two sets of transformation rules (T1 and T2). M1 and M2 can be conceptual, logical or physical metamodels. All possible combinations of roundtrip scenarios are shown in Figure 2.

| M1 \ M2 | **Conceptual** | **Logical** | **Physical** |
|---|---|---|---|
| **Conceptual** | Case 1: Translation | Case 2: Forward Engineering | *Case 3: Forward Engineering* |
| **Logical** | Case 4: Reverse Engineering | Case 5: Migration | Case 6: Forward Engineering |
| **Physical** | *Case 7: Reverse Engineering* | Case 8: Reverse Engineering | Case 9: Migration |

*Figure 2: Characterizing roundtrip steps*

*Forward engineering* occurs each time M2 has a lower level of abstraction than M1. Three different cases are to be considered:

- Case 2: M1 and M2 are respectively conceptual and logical metamodels.
- Case 3: M1 and M2 are respectively conceptual and physical metamodels. This is a case of forward engineering. However skipping the logical intermediate level can cause errors and make maintenance of the resulting systems more difficult.
- Case 6: M1 and M2 are respectively logical and physical metamodels.

*Reverse engineering* is encountered when M2 has a higher level of abstraction than M1. Three different cases occur:

- Case 8: M1 and M2 are respectively physical and logical metamodels.
- Case 7: M1 and M2 are respectively physical and conceptual metamodels. Although it is undesirable to do so, it is nonetheless a case of a possible reverse engineering. Once again, skipping the logical step can lead to inconsistent results.
- Case 4: M1 and M2 are respectively logical and conceptual metamodels.

*Mapping* a conceptual metamodel M1 into a conceptual metamodel M2 is required when two different conceptual formalisms coexist (Case 1). This the case when, for example, M1 is an Extended Entity Relationship metamodel and M2 is a UML metamodel. When both formalisms are equally expressive, T1 and T2 are reversible transformation rules. This mapping is classical in database engineering and does not present any specific challenge in the context of NoSQL systems.

Finally, two cases deal with *migration* of databases, either at a logical level (Case 5) or at a physical level (Case 9). For example, migrating from Neo4j to OrientDB is an example of physical migration whereas migrating from graph database to relational database illustrates a logical migration.

Roundtrip engineering may encompass several combinations of the cases described above. The generic roundtrip engineering problem may be reduced to the paths described at Figure 3, without loss of generality. The process also contains enrichment of models due to new requirements and/or reengineering of models.

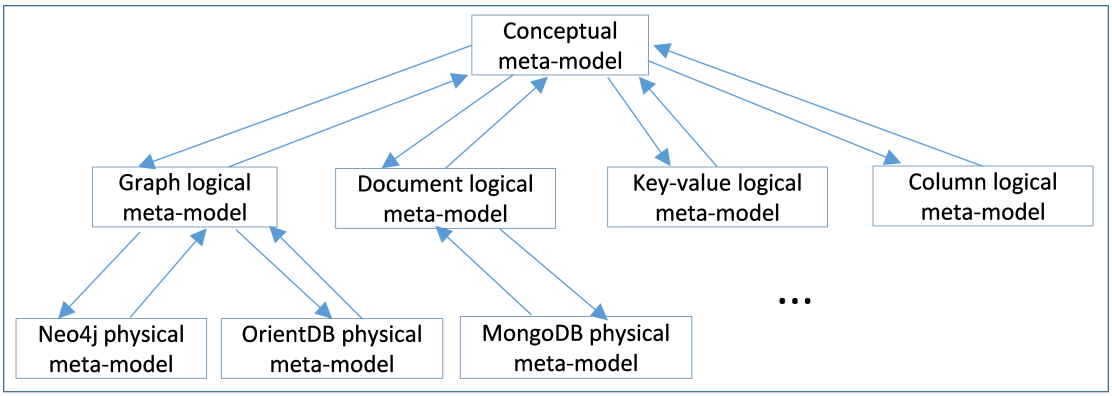



*Figure 3: Recommended paths for NoSQL database roundtrip engineering*

Each arrow refers to either Case 2, Case 4, Case 6, or Case 8. We don't represent the arrows corresponding to Case 1 since it is not specific to NoSQL systems. We also don't take into account Cases 3 and 7 since they skip the logical level, which is not considered as a good practice. Finally, we don't represent Cases 5 and 9 since we recommend always to maintain the consistency between levels. Migrating from a logical model to another one requires defining a conceptual intermediate step. In the same way, migrating from a physical model to another one requires at least defining a logical intermediate step and, if needed, a subsequent conceptual step.

Even if there is no roundtrip engineering approach for NoSQL databases, nevertheless the literature contains several contributions corresponding either to one arrow of Figure 3 or to two consecutive arrows constituting an acyclic path. Moreover, existing approaches only define transformation rules but don't take into account the additional information required to anticipate the roundtrip paths.

## 4 Illustrative scenario

In this section, we illustrate our framework with an example. Let's consider the following conceptual model representing information about publications (Fig. 4), taken from (Akoka et al. 2017). Entities contain information about scientific papers, their sources (journals, conferences), their authors, and their affiliations. Let us note the reflexive citation relationship between papers. Terms are keywords characterizing papers. We also added a reviewer entity and a researcher entity. Author and reviewer are subtypes of researcher entity. We suppose that an author may have several affiliations but when he/she publishes a paper, he/she has to declare a unique affiliation. Thus, we represent a ternary relationship between papers, authors, and affiliations.

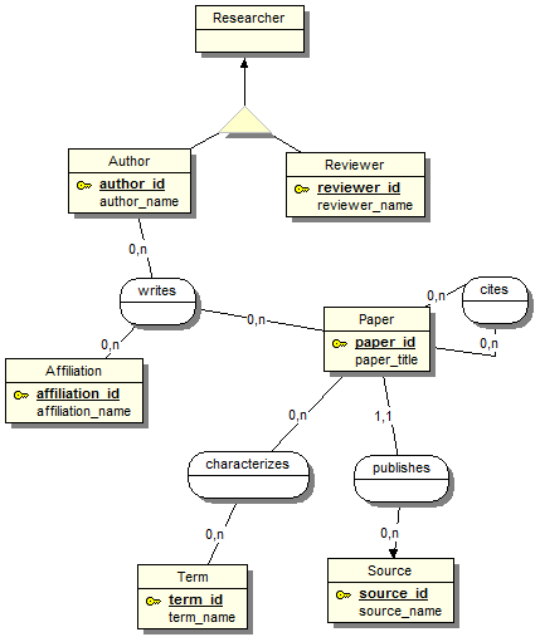



*Figure 4: Example of conceptual model (Akoka et al. 2017)*

A first step performing a forward engineering process is described as follows. A set of transformation rules (described in (Akoka et al. 2017)) allowed us to generate a logical property graph database (Figure 5). As an example of a rule, the ternary relationship *writes* in the conceptual model becomes a node of the graph. This node is connected to the three nodes resulting from the transformation of the three entities involved in the relationship. As an additional information, our approach provides also the logical graph with its estimated size (volume attribute). Another set of rules leads to a physical Neo4j graph. In addition, the information regarding the size enables the generation of a Neo4j test database containing as many nodes and edges as estimated by the designer.

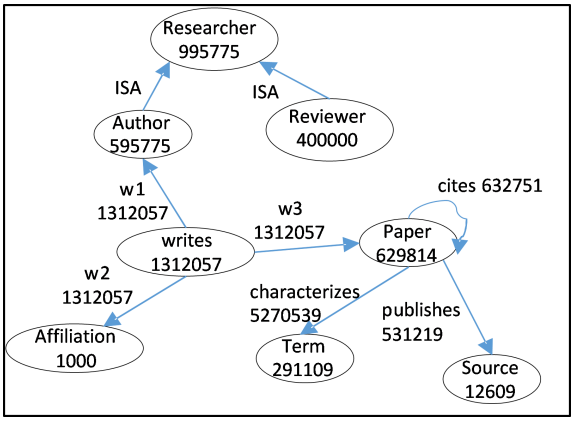



*Figure 5: Resulting graph logical model (Akoka et al. 2017)*

Let us suppose that the database administrator received additional data describing papers such as year of publication, number of pages, and language. Through a reverse engineering process, this information is propagated into the logical graph and to the subsequent conceptual model. We performed this reverse process by applying the set of rules described in (Comyn-Wattiau and Akoka 2017). The final result is presented at Figure 6.

Let us consider that the end users then asked to enrich the database with historical information on past affiliations of researchers. This led us to the model of Figure 7 where the relationship *mentions* links researchers, affiliations, and dates. A new forward engineering process containing two steps allowed us to generate the updated Neo4j database.

Finally, let us suppose that the end users would like also to automatically access not only to the information on papers but also to the full text papers. Neo4j is not able to store documents. Thus the database administrator proposes to migrate the database toward an OrientDB environment. OrientDB combines graph and document logical models. The resulting logical schema is presented at Figure 8. At the physical level, the nodes representing paper information are linked to documents storing full-text papers, but this is not visible at the logical level. Moreover, to maintain the consistency with different modeling levels, we have to generate the updated conceptual model of Figure 9 mentioning the full-text attribute.

We summarize the main steps of our illustrative scenario at Figure 10. It consists mainly of successive steps of model enrichment, forward engineering, and reverse engineering steps. Let's compare the initial conceptual model of Figure 4 and the final one of Figure 9. The different enrichments allowed us to complete the description of conceptual objects (entities or relationships) and/or to add new objects. Moreover, the successive execution of forward and reverse engineering processes are not all inverse transformations. In our example, the main difference lies in the ternary *writes* relationship which is transformed into a node and, conversely, becomes an entity with three binary relationships respectively with author, affiliation and paper entities, which is less expressive. A round-trip engineering process must be able to keep trace of the forward transformation in order to be able to reverse it. If this trace mechanism is not available, a quality analysis of the conceptual model generated through the reverse process should be able to detect the semantic poverty of the *writes* entity and its associated relationships *w1*, *w2*, *w3*. This quality analysis may suggest to the designer to provide a better naming of such objects.

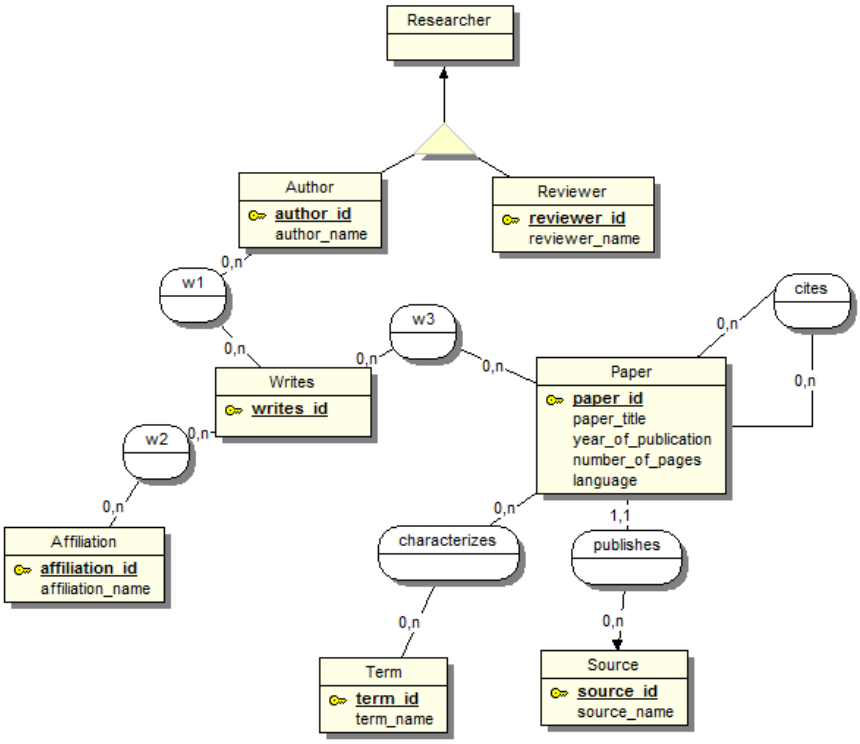



*Figure 6: New conceptual model (reversed)*

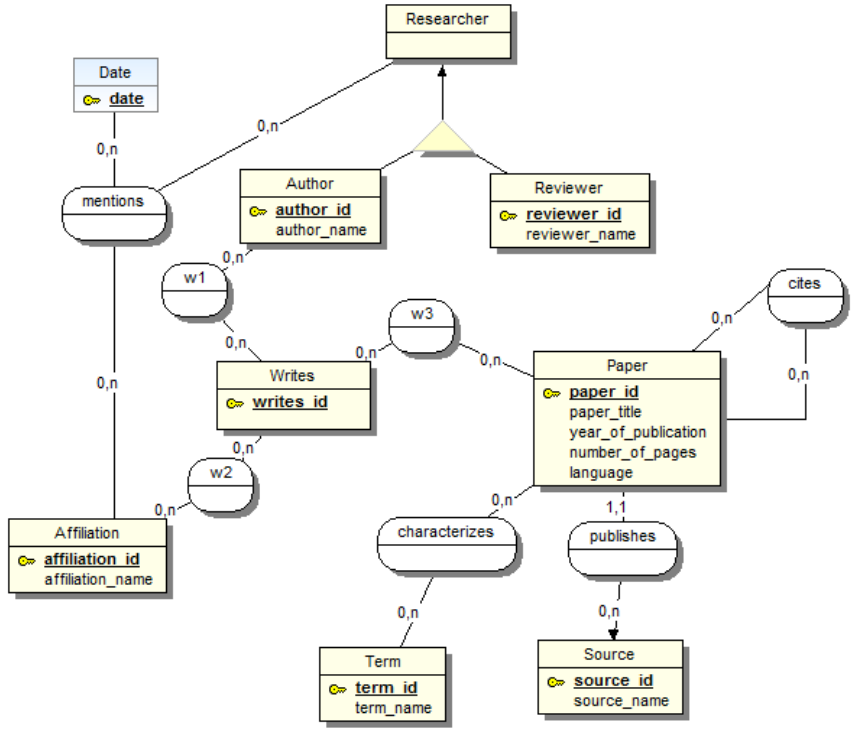



*Figure 7: Enriched conceptual model*

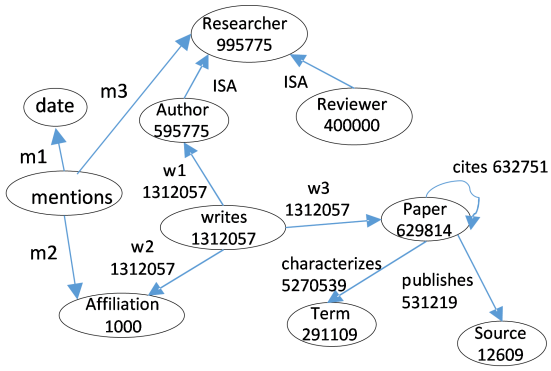



*Figure 8: Updated logical graph model*

## 5 The resulting framework

Like any software system, NoSQL databases require maintenance efforts. Software maintenance usually is categorized into four types. Corrective maintenance is dedicated to fix problems. Adaptive maintenance includes modifications applied to keep the software up-to-date and aligned with its environment. Perfective maintenance takes into account new user requirements. Finally preventive maintenance occurs when organizations anticipate future problems.

*Corrective maintenance* impacts the physical level of the database. Such modification must be propagated at both the logical and the conceptual levels through a reverse process. At each abstraction level, a quality analysis may lead to a roundtrip engineering path composed successively of Case 8, Case 4, Case 2, and Case 6.

*Adaptive maintenance* may, for example, lead to the migration of a database due to a new release of its platform. In such a case, the editors generally ensure an ascending compatibility, limiting the impact to the physical level.

*Perfective maintenance* occurs each time new user requirements have to be considered. It should be a classical forward engineering process propagating the new requirements from the conceptual level to the logical and physical levels. However, if the only artifact is the code and/or if previous changes have been integrated at the physical level without performing a backward propagation, a roundtrip engineering path composed successively of Case 8, Case 4, Case 2, and Case 6 must be executed.

*Preventive maintenance* takes place in many companies where relational databases reach their limits in their capability to meet the volume and/or variety requirements. A migration to a NoSQL system (Case 5) is necessary. In such a situation, the recommended path is composed successively of Case 4, Case 2, and Case 6. Another preventive maintenance happens when non-functional requirements (security, performance, etc.) are no longer met. A migration to another physical environment (Case 9) may be the solution. A path composed successively of Case 8 and Case 6 is recommended.

Roundtrip engineering systematically propagates changes forward or backward. Although this appears to be an additional task, it enables anticipating and facilitating any changes that the maintenance of the system, whatever it may be (corrective, adaptive, perfective, or preventive), requires.

Implementing a roundtrip engineering process therefore requires the design of a knowledge base that combines rule sets, trace templates, and quality assessment models. Some of the latter have been developed for relational databases. Much remains to be done in the field of NoSQL databases. The framework sketched at Figure 11 summarizes our approach.

The first step consists in taking into account maintenance demands and qualifying them. A guiding step then recommends a RTE scenario. The latter is implemented using the knowledge base. It is followed by a quality analysis allowing the software engineer to commit the changes or to reiterate the process.

## 6 Conclusion and further research

Roundtrip engineering allows us to perform the transformations imposed by different types of software maintenance. In this paper, we studied the case of NoSQL database maintenance. We

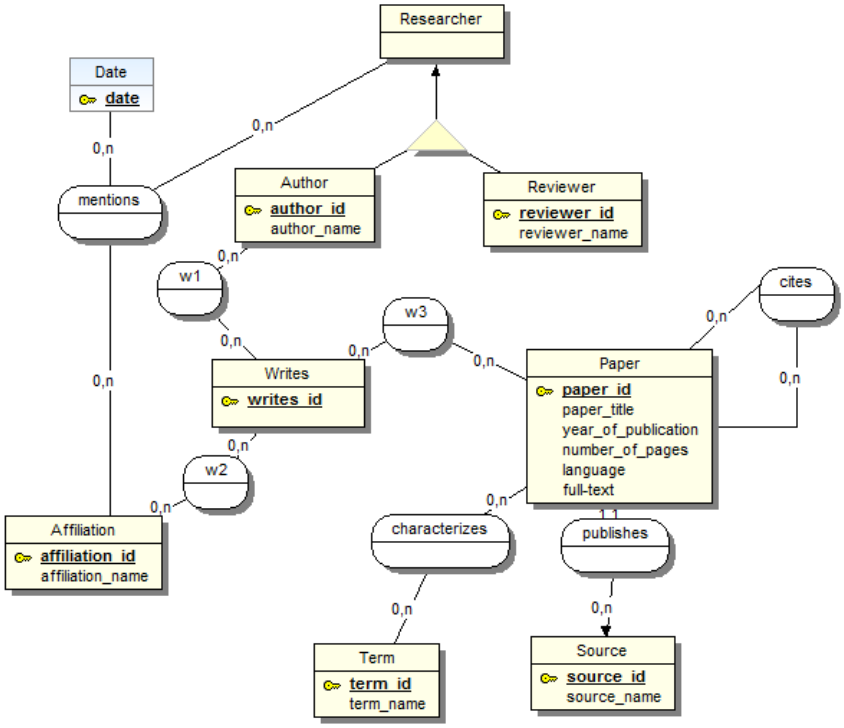



*Figure 9: Final conceptual model*

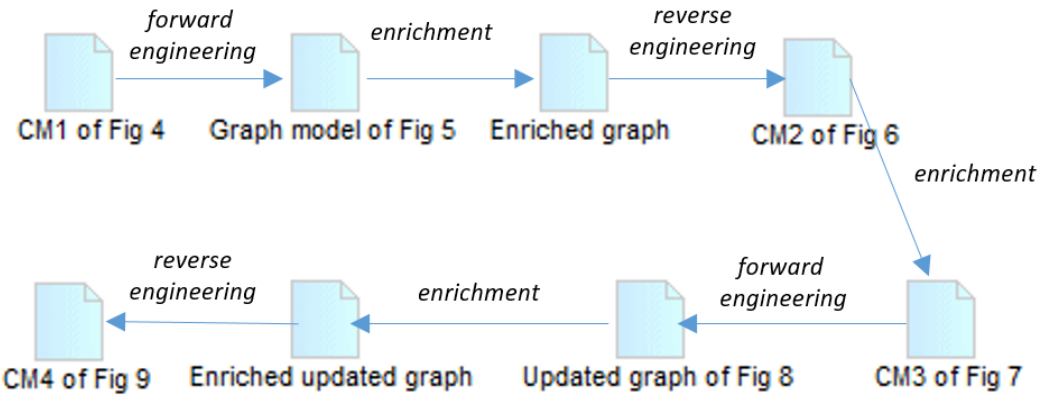



*Figure 10: The illustrative process*

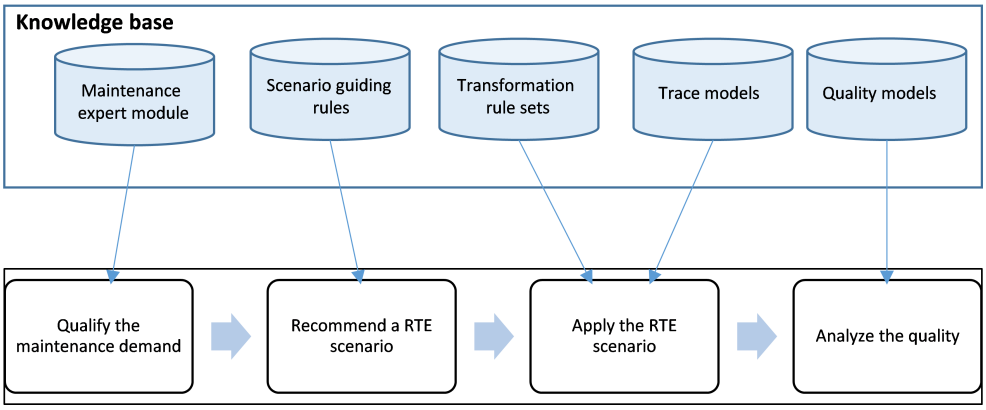



*Figure 11: Final conceptual model*

proposed a framework dedicated to roundtrip engineering of NoSQL databases. The framework encompasses a knowledge base and a guiding process. The knowledge base is composed of four main modules: a maintenance expert module, a set of guiding rules, several transformation rule sets, trace models, and quality evaluation models. The RTE process starts by qualifying the maintenance demand, then recommends a scenario, applies it and evaluates the resulting quality.

Starting from a roundtrip generic scenario, we propose several roundtrip scenarios combining forward and reverse engineering processes. We demonstrate the feasibility of our approach with an illustrative scenario related to a property graph database. The illustrative scenario consists of successive steps of model enrichment combined with forward and reverse engineering processes.

Future research will consist in designing and implementing the main components of the knowledge base. In order to validate the approach, we will test the resulting prototype with several case studies. Some components of the framework will reuse the results obtained for relational database systems while others will be created ex nihilo.

*We dedicate this article to Heinrich Mayr for his fruitful contributions to conceptual modeling.*

## References

Akoka J., Comyn-Wattiau I., Prat N. (2017) A Four V's Design Approach of NoSQL Graph Databases In: Advances in Conceptual Modeling: ER 2017 Workshops AHA, MoBiD, MREBA, OntoCom, and QMMQ de Cesare S., Frank U. (eds.) Springer International Publishing, pp. 58–68 https://doi.org/10.1007/978-3-319-70625-2_6

AndroMDA. http://www.andromda.org/. Last Access: Accessed: 2017-12-26

Angyal L., Lengyel L., Charaf H. (2008) A Synchronizing Technique for Syntactic Model-Code Round-Trip Engineering. In: 15th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems (ecbs 2008), pp. 463–472

Antkiewicz M., Czarnecki K. (2006) Framework-Specific Modeling Languages with Round-trip Engineering. In: Proceedings of the 9th International Conference on Model Driven Engineering Languages and Systems. MoDELS'06. Springer-Verlag, Genova, Italy, pp. 692–706 http://dx.doi.org/10.1007/11880240_48

Aßmann U. (2003) Automatic Roundtrip Engineering. In: Electronic Notes in Theoretical Computer Science 82(5) SC 2003, Workshop on Software Composition (Satellite Event for ETAPS 2003), pp. 33–41

Boger M., Graß E., Köster M. (2007) Poseidon for UML Users Guide

Booch G., Rumbaugh J., Jacobson I. (1998) The Unified Modeling Language User Guide. Addison-Wesley

Bork M., Geiger L., Schneider C., Zündorf A. (2008) Towards Roundtrip Engineering - A Template-Based Reverse Engineering Approach In: Model Driven Architecture – Foundations and Applications: 4th European Conference, ECMDA-FA 2008 Schieferdecker I., Hartman A. (eds.) Springer Berlin Heidelberg, pp. 33–47 https://doi.org/10.1007/978-3-540-69100-6_3

Borland Software Solutions (2009) Borland Together

Buchmann T., Westfechtel B. (2013) Towards Incremental Round-Trip Engineering Using Model Transformations. In: 39th Euromicro Conference on Software Engineering and Advanced Applications, pp. 130–133

Ciccozzi F., Cicchetti A., Sjodin M. (2011) Towards a Round-Trip Support for Model-Driven Engineering of Embedded Systems. In: 37th EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 200–208

Comyn-Wattiau I., Akoka J. (2017) Model-Driven Reverse Engineering of NoSQL Property Graph Databases. In: Big Data 2017: IEEE International Conference on Big Data. IEEE, pp. 453–457

Czarnecki K., Helsen S. (2003) Classification of Model Transformation Approaches. In: Proceedings of the 2nd OOPSLA Workshop on Generative Techniques in the Context of the Model Driven Architecture, USA

Czarnecki K., Helsen S. (2006) Feature-based survey of model transformation approaches. In: IBM Systems Journal 45(3), pp. 621–645

Demeyer S., Ducasse S., Tichelaar S. (1999) UML shortcoming for coping with round-trip engineering. In: UML'99 Conference Proceedings. Springer-Verlag

Engels G., Küster J. M., Heckel R., Groenewegen L. (2001) A Methodology for Specifying and Analyzing Consistency of Object-oriented Behavioral Models. In: Proceedings of the 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-9). ACM, Vienna, Austria, pp. 186–195 http://doi.acm.org/10.1145/503209.503235

Favre J. M. (2004) CaCOphoNy: metamodel-driven software architecture reconstruction. In: 11th Working Conference on Reverse Engineering, pp. 204–213

Greiner S., Buchmann T., Westfechtel B. (2016) Bidirectional transformations with QVT-R: A case study in round-trip engineering UML class models and java source code. In: 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), pp. 15–27

Hailpern B., Tarr P. (2006) Model-driven development: The good, the bad, and the ugly. In: IBM Systems Journal 45(3), pp. 451–461

Hettel T., Lawley M., Raymond K. (2009) Towards Model Round-Trip Engineering: An Abductive Approach In: Theory and Practice of Model Transformations: Second International Conference (ICMT 2009) Paige R. F. (ed.) Springer Berlin Heidelberg, pp. 100–115 https://doi.org/10.1007/978-3-642-02408-5_8

Ivkovic I., Kontogiannis K. (2004) Tracing evolution changes of software artifacts through model synchronization. In: 20th IEEE International Conference on Software Maintenance. IEEE, pp. 252–261

Kellokoski P. (2000) Round-trip Engineering. MA thesis, University of Tampere, Finland

Kleppe A. G., Warmer J., Bast W. (2003) MDA Explained: The Model Driven Architecture: Practice and Promise. Addison-Wesley Longman Publishing Co., Inc.

Küster J. (2004) Consistency Management of Object-Oriented Behavioral Models. PhD thesis, University of Paderborn, Germany, pp. 1–306

Larrson H., Burbeck K. (2003) CODEX – An Automatic Model View Controller Engineering System.. Proceedings of Workshop Model Driven Architecture, Foundations and Applications, CTIT Technical Report TR-CTIT—03-27, University of Twente

Macedo N., Cunha A. (2016) Least-change bidirectional model transformation with QVT-R and ATL. In: Software & Systems Modeling 15(3), pp. 783–810 https://doi.org/10.1007/s10270-014-0437-x

Mayr H. C., Michael J., Ranasinghe S., Shekhovtsov V. A., Steinberger C. (2017) Model Centered Architecture In: Conceptual Modeling Perspectives Cabot J., Gómez C., Pastor O., Sancho M. R., Teniente E. (eds.) Springer International Publishing, pp. 85–104 https://doi.org/10.1007/978-3-319-67271-7_7

Nagowah L., Goolfee Z., Bergue C. (2013) RTET - A round trip engineering tool. In: International Conference of Information and Communication Technology (ICoICT), pp. 381–387

Nickel U., Niere J., Wadsack J., Zündorf A. (2000) Roundtrip Engineering with FUJABA. In: Ebert J., Kullbach B., Lehner F. (eds.) Proceedings of 2nd Workshop on Software-Reengineering (WSR), Bad Honnef, Germany, Fachberichte Informatik, Universität Koblenz-Landau

Odutola K., Oguntimehin A., Tolke L., Wulp M. (2001) ArgoUML Quick Guide

Orshalick J., Assar N. (2010) JBoss Seam: Agile Ria Development Framework, Red Hat Inc

Rational Software (2006) IBM Rational Rose – Data Sheet

Ruiz F. J. B., Molina J. G., García O. D. (2017) On the application of model-driven engineering in data reengineering. In: Information Systems 72(Supplement C), pp. 136–160

Sendall S., Kozaczynski W. (2003) Model transformation: the heart and soul of model-driven software development. In: IEEE Software 20(5), pp. 42–45

Winterfeldt D. (2012) JSpring by Example, Version 1.2.1

Zhang J., Lin Y., Gray J. (2005) Generic and Domain-Specific Model Refactoring Using a Model Transformation Engine In: Model-Driven Software Development Beydeda S., Book M., Gruhn V. (eds.) Springer Berlin Heidelberg, pp. 199–217 https://doi.org/10.1007/3-540-28554-7_9

# Understanding *Semantic Completeness* in Rule Frameworks for Modeling Cardinality Constraints

Faiz Currim[a], Sudha Ram[*,a]

[a] Department of Management Information Systems, Eller College of Management, University of Arizona, Tucson, USA

Abstract. *Modeling organizational rules during conceptual design provides a more accurate picture of the underlying domain and helps enforce data integrity. In a database development context, there are many advantages to explicitly representing rules during conceptual design. Early modeling ensures they are visible to designers and users, thus improving requirements and validation. The rules can then be semi-automatically translated into logical design code. One limitation to widespread adoption of such modeling is variance in standards and semantics of rules. We consider cardinality constraints—a useful and integral part of conceptual database design. Many papers discussing classification frameworks for cardinality exist. Completeness of such schemes has always been in question since well-defined criteria do not exist to evaluate them. We suggest a "reverse engineering" approach, i.e., one of defining conceptual modeling constraint completeness based on mappings from the relational model. We develop a correspondence from relational algebra operator combinations to existing semantic constraint types. In doing so, we also come up with a new category of set-level cardinality constraints not previously examined in literature. We believe our work demonstrates a unique approach to establishing conceptual framework completeness and enables standardization of rule semantics which in turn allows for semantics-based (as opposed to procedural-based) representation. On the implementation side, it supports developing automated mechanisms for translating constraints to improve developer productivity.*

Keywords. Database design • Conceptual database modeling • Cardinality constraints

## 1 Introduction and Motivation

Cardinality constraints have long been an integral part of conceptual database diagrams since the original entity-relationship (ER) model proposed by Chen (Chen 1976). Other conceptual modeling standards including Unified Modeling Language (UML) (OMG 2015), and Object-role modeling (ORM) (Halpin and Morgan 2010) also provide support for cardinality. As do pre-design models such as KCPM (Vöhringer and Mayr 2006). (The terminology may vary across models, e.g., UML or KCPM may term it as multiplicity.) A variety of papers have examined cardinality constraints in detail, and many frameworks and taxonomies have been proposed to comprehensively organize the types of cardinality constraints (Lenzerini and Santucci 1983; Thalheim 1992; Liddle et al. 1993; McAllister 1998; Ram and Khatri 2005). With any taxonomy, including one for cardinality, the question of semantic *expressiveness* or *completeness*[1] (Navathe et al. 1992) is pertinent. Establishing completeness is valuable from a standpoint of both theory and practice, and allows one to establish an exhaustive mapping into implemented database constraints. Though authors have

* Corresponding author.
E-mail. ram@eller.arizona.edu
We thank Nicholas Neidig, Alankar Kampoowale, Girish Mhatre, Anish Padiyara and Mark Vanderflugt for assistance with developing the CARD system prototype.

[1] We use the terms completeness, comprehensiveness and expressiveness interchangeably. Some writers prefer the term expressiveness since completeness often implies an absolute completeness which can be difficult to establish.

sought to address the issue of completeness of their frameworks, it has remained a difficult and open question. This is because it is hard to prove in advance that a scheme (e.g., conceptual model or a rule framework) is fully expressive. Options to demonstrate completeness include envisioning a large number of possible scenarios (including covering what similar models have used in the past) or extensive testing in the field. Each has its costs and drawbacks, can be time-consuming, and remain susceptible to not covering the needs of a new application. This can lead to long delays in adoption of a framework, since it may be impeded due to the perception of it being "yet another incremental version" that augments expressiveness by some small amount but is not complete.

Proving expressiveness in the context of rule frameworks is important, as it allows for a comprehensive set of constraint specifications and resultant translation into code. This in turn provides a much stronger case for incorporating rule frameworks into design tools. Having a visible rule repository is important for good managerial decisions (Von Halle 2001) and researchers and practitioners have recommended that user-specified business rules applicable to data should be documented in the database schema itself (ISO 1987; Simsion 2001). In data modeling, cardinality is one such class of rules that must be modeled.

In this work, we focus on cardinality rules.[2] Our objective is to determine how completeness can be established for cardinality constraint frameworks. We discuss previous approaches to proposing comprehensiveness in the conceptual modeling domain, and define completeness in a novel way. Instead of viewing the transition from the conceptual to the logical design stage as a one-way street, we "reverse map" from relational algebra to conceptual modeling constraint kinds. We consider combinations of algebraic operations, and show that a complete framework must express constraints corresponding to all relevant operation arrangements. We also test the feasibility of such mapping with a proof-of-concept prototype system and present some findings and recommendations for DBMS support of these kinds of rules.

The rest of our paper is structured as follows. We begin by reviewing prior work in rule representation in conceptual and logical design in section 2. In section 3, we address the issue of completeness for semantic modeling and why our proposed approach is reasonable. Section 4, contains the discussion of the various relational algebra operators and how different combinations of them map to cardinality constraints (with a SQL mapping for each constraint kind). Thereafter, we discuss our evaluation and present our recommendations in section 5. Section 6 contains the conclusions and suggestions for future work.

## 2 Review of Related Work

We see support for representing a variety of rule types in conceptual database design. The original ER proposal allowed for representing identifying attributes[3] , cardinality, and implicitly—referential integrity through the specification of relationships. Extensions to the ER model have allowed designation of the mandatory vs. optional properties for relationship participation, as well as for attributes, i.e., whether they can contain null values (Figure 1). Newer modeling tools may further allow specification of additional constraints such as data types and domain ranges (see Figure 2) by providing an interface that bridges conceptual and logical design.

From a data management perspective, such rules function as integrity constraints on a database helping to ensure that the business policies and semantics of the application are incorporated into the database (Storey et al. 1996; Date 2000a; Simsion

[2] We use the terms *constraints* and *rules* synonymously though we recognize that there are multiple interpretations for "rule" in literature. We use *policy* to refer to the underlying business directive that guides the constraint.

[3] Some authors prefer the term "primary key"; however, the notion of primary keys comes from the relational model (logical design), and we adopt the usage of "identifying attributes" in context of conceptual design.

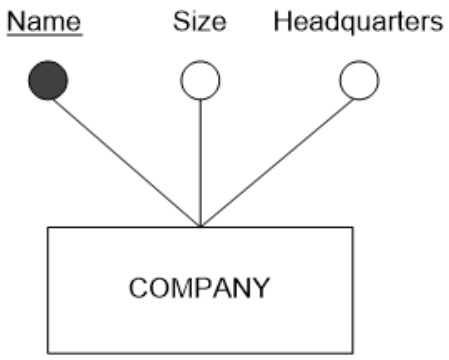



*Figure 1: In this diagram, the attributes* size *and* headquarters *may contain null values (syntax from (Umanath and Scamell 2007))*

2001). All well-known conceptual grammars[4] provide support for cardinality, as it is needed to determine the translation of a conceptual schema into the corresponding logical relations (for the purposes of our paper we assume the relational model is used during logical design). Similarly, cardinality is employed when mapping from a pre-design glossary to a conceptual schema (Mayr and Kop 2002). Cardinality rules are also useful for other database purposes including normalization (Navathe et al. 1992), schema integration (Ramesh and Ram 1997), query optimization (Thalheim 1996), and managing privacy for sensitive data (Sweeney 2002).

The importance of capturing and visibly representing business rules has been highlighted by a number of efforts (Date 2000b; Hay et al. 2000; Von Halle 2001; Ross 2005; OMG 2017).

However, inclusion of support for representing cardinality in modeling methodologies and tools has been somewhat limited. One of the issues with cardinality is the complexity in its semantics when generalized beyond a binary relationship case or when dealing with temporal or spatial data. Compounding this problem is the variations in intended semantics as used in different modeling grammars (Ferg 1991; Liddle et al. 1993). Over time, different approaches to address the problem have been proposed. UML, for example, allows the specification of complex cardinality constraints using the Object Constraint Language (OCL) (OCL 2014). Adapting an example from Warmer and Kleppe (J. Warmer and A. Kleppe 1999), consider the SubmitBids relationship schema in Figure 3. A requirement that "any supplier and project combination can appear in the SubmitBids relationship between [*h* .. *k*] times", where *h* and *k* are a user-specified integer bounds, can be specified using the OCL code in Figure 4.

A similar approach can be adopted to assert, for instance, that a student can take the same course up to *m* times (across semesters) or an employee may be assigned to work on the same project up to *n* times in a year. Being a procedural specification in OCL, this option is not without its own deficiencies. It does not scale well, and each time we see such a constraint—we must rewrite the logic for its implementation.

An alternative approach that exists in literature, is to specify a conceptual taxonomy of the various cardinality constraints, and then take advantage of the classification scheme to denote the constraint type and its parameters. Using a syntax formalized previously (Currim and Ram 2012), we can state the earlier participation[5] constraint on the SubmitBids relationship by:

```
CARD-R-PT (SubmitBids, (SUPPLIERS,
PROJECTS) ) IN [h:k]
```

Thus, conciseness of annotation is better achieved using a semantics-based or conceptual classification. This can lead to improved analyst productivity, and reduce the chance of errors while writing a program. An advantage of this method is that it can be mapped to the corresponding OCL

[4] We use the term *grammar* to refer to the formalism specifying the constructs and rules for conceptual design, e.g., ER Model. We use the term *schema* (or *script*) to refer to the abstract description of the real world developed using the constructs provided by the *grammar* (Wand and Weber 2002).

[5] Briefly explaining the syntax, in their scheme CARD-R-PT stands for a cardinality (CARD) constraint on an interaction relationship (R), restricting participation (PT). The parameters are the relationship involved, and the constrained entity classes.

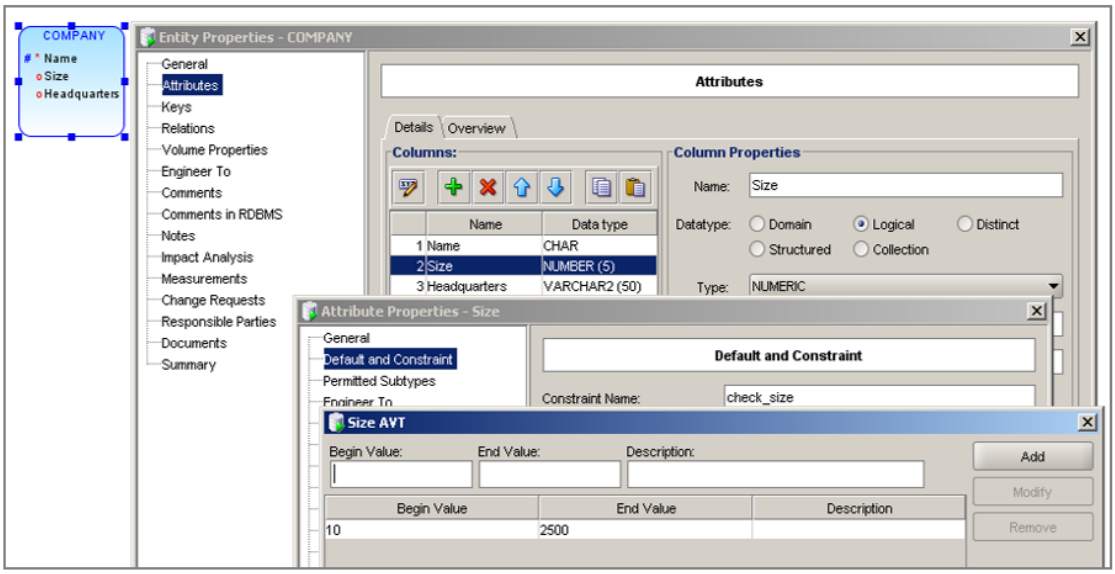



*Figure 2: Using a conceptual design tool to specify the data type for the Size attribute, and a check constraint to be implemented upon conversion to relational table (i.e., logical-level) creation code*

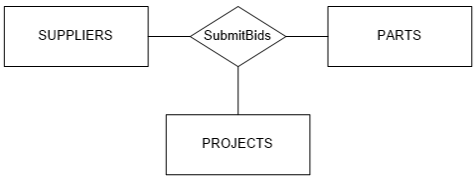



*Figure 3: The Submit Bids relationship*

specification or triggers (which we show later in this paper) in a straightforward manner. On the other hand, a limitation with this approach, as demonstrated by the augmented taxonomies published over the years (Lenzerini and Santucci 1983; Thalheim 1992; Liddle et al. 1993; McAllister 1998; Ram and Khatri 2005), is that establishing comprehensiveness in the organized types of cardinality constraints is difficult. Successive efforts have added support for new kinds of constraints. While this improves the body of knowledge, it is still desirable to have a sense of completeness so that modeling tools can incorporate the taxonomy and the related translation plans. Otherwise, an argument can be made that logical-level proposals that generate pseudo-code, while less efficient and insensitive to the underlying semantics, provide

```
sb-s: SubmitBids -> SUPPLIERS
sb-j: SubmitBids -> PROJECTS

SUPPLIERS -> forAll( s |
  PROJECTS -> forAll ( j |
    SubmitBids -> select (sb |
    sb.sb-s = s and sb.sb-j = j
    ) -> size >= h
    and
    SubmitBids -> select (sb |
    sb.sb-s = s and sb.sb-j = j
    ) -> size <= k
  )
)
```

*Figure 4: Sample OCL code for specifying a participation constraint*

the flexibility of incorporating new kinds of constraints. To enable wider adoption of taxonomic cardinality specification approaches for rule processing and service-oriented computing in a heterogeneous environment, we feel a standardized and *complete* framework for cardinality constraints is important.

## 3 Addressing Completeness

As mentioned earlier, addressing the question of rule framework completeness is valuable from both a research and business standpoint. In the context of semantic modeling and cardinality, while authors have sought to address the matter of completeness of their frameworks, it has remained an open issue. Completeness is difficult to measure because there is no easy way to establish a priori that a model has the necessary constructs to capture the semantics of every possible application. The task can be simplified somewhat by narrowing the scope of the taxonomy or model (which is one of the reasons, besides compactness of representation, that we see a proliferation of domain-specific conceptual grammars). Having reduced the target modeling space, one can test completeness by undertaking multiple case studies in a variety of organizations in the field. The greater the number of studies, the more confidence one has in the expressiveness of the taxonomy.

Since grammars like the Entity-Relationship model have already been extensively field-tested, some authors have adopted the tactic of measuring *relative completeness* where the new model is measured against existing grammars (Bajaj and Ram 2002). Thus, a conceptual model can be considered relatively complete if its constructs are at least as expressive as previously developed models. Applying the norm to constraint frameworks, we can say that a framework is relatively complete if the classified constraints incorporate those seen in existing constraint systems. Most work to date has implicitly adopted this approach while demonstrating that their proposed framework encompasses previous classificatory schemes (Liddle et al. 1993; McAllister 1998; Ram and Khatri 2005). An argument of insufficient confidence could, however, be made against this methodology since cardinality rules are not always thoroughly specified while developing a conceptual schema, and thus our faith in the completeness of existing classification mechanisms may not be strong.

To address this issue we asked ourselves, "What alternative benchmarks could be used for expressiveness?" There was no simple answer due to the absence of pre-existing criteria to define completeness. We knew that extensive organizational testing is not practical in a reasonable timeframe. The approach we chose instead was to adapt our test for completeness by taking advantage of existing work in relational query sub-languages (in our case, relational algebra), where completeness has already been well-defined.

Previous efforts on completeness for ER-based query languages (Atzeni and Chen 1981; Campbell et al. 1985) are not based on relational algebra or relational calculus, and instead have defined expressiveness based on constructs within the ER model. Our approach diverges from this, and we argue that our "reverse engineering" (working backwards from the logical design) strategy is acceptable for three reasons. Firstly, the underlying theoretical model for both the entity relationship and the relational models is the same, i.e., set theory. The constructs for capturing and storing data can be visualized as a derivation of sets in both models. This, in part, accounts for the well-defined and straightforward mapping between the two models. Secondly, since relational language completeness has been well-tested (and can thus be construed as a sound measure), we may as well take advantage of this knowledge while formulating a measure of completeness for constraints. Finally, we feel that since the implementation of conceptual model cardinality constraints will typically be done in a relational database, it would be reasonable to think that a classification scheme is complete only if every cardinality constraint type that can be implemented in a relational algebra is also available in a conceptual model cardinality framework.

## 4 Establishing Completeness Using Relational Algebra

To begin, we intuitively establish the correspondence between constraints and queries. Let us take the example of a constraint: *An employee can work for between 1 and 25 projects* specified on the work_on relationship (see Figure 5). In the relational model, this relationship would map to a table work_on and an SQL query would be executed to perform a count of projects for the employee in the modified tuple. Depending on the results, the system could determine if the constraint were satisfied or violated.

Correspondingly, we argue that the evaluation of every cardinality constraint can be mapped to an SQL query (or relational algebra expression) to be checked upon a database operation (insert, delete or update). Since the query to evaluate a cardinality constraint is but one kind of query, the set of all possible cardinality constraint queries is a strict subset of all possible queries. Thus, a language that is constraint-complete will be a subset of a language that is relationally complete. Next, we discuss the operators required for relational completeness, and the relevant subset needed for checking cardinality constraints.

For a language to be relationally complete, it must be capable of the following relational operations (Codd 1972): projection ($\pi$), selection ($\sigma$), Cartesian product ($\times$), union ($\cup$), and set difference ($-$). Further, we consider a common extension, that of supporting aggregation and grouping ($\Im$) and including functions (like counts) on attributes to be performed in a projection. As might be expected, we do not need the full functionality of all the relational operators $\pi$, $\sigma$, $-$, $\Im$, $\times$, $\cup$ for checking cardinality constraints. We elaborate below.

The projection $\pi$ operator is unique in that it filters attributes rather than tuples (or in other words: extracts properties rather than entities). In a general query language, any combination of attributes and expressions may be desired in the output and therefore the use of the $\pi$ operator is diverse. However, for checking a cardinality constraint—the only kind of query we are interested in is a count. We denote the count based on a projection with the symbol $\pi^{c}$. Two different kinds of counting may be performed. The simplest is to count entire tuples (SELECT COUNT (*) in SQL), which we denote by $\pi^{c^*}$. Alternatively, a count of distinct attribute values may be carried out (SELECT COUNT (DISTINCT ⟨attribute⟩)), which we denote using $\pi^{cA}$.

The operators of Union and Set Difference $\{\cup, -\}$ serve to either augment or reduce the membership of the set under consideration (leading to the creation of a new set). They are similar in that they change the set membership, but not how elements are counted within a defined set and therefore do not change the nature of the cardinality constraint being considered. Thus, they are not as important from the perspective of a cardinality framework.

The Cartesian product $\{\times\}$ is usually performed in conjunction with the selection operator $\sigma$ to create a "join". Conceptual model constraints often implicitly assume the use of joins. For example, the constraint: *An employee with the designation of "Senior Consultant" can work for between 1 and 10 projects*, defined on the work_on relationship (Figure 5), contains a predicate on the participating EMPLOYEES entity class, and hence implicitly uses a join (when considered in the relational model). The constraint frameworks we have encountered thus far only consider joins within a relationship's participating entity classes, rather than joins among classes that may span a chain of relationships (i.e., they would not consider a constraint defined between employees and clients in Figure 6). To address this, we propose the notion of *virtual relationships*, discussed in more detail elsewhere (Currim 2004); which naturally maps to the definition of a logical view. A point to note is that the *nature of counting* that is performed *does not change*, merely the scope. Thus, we can continue our analysis excluding such joins without loss of generality.

Finally, we consider the two operators that when applied to a set allow us to count specific members or aspects of that set. Both selection

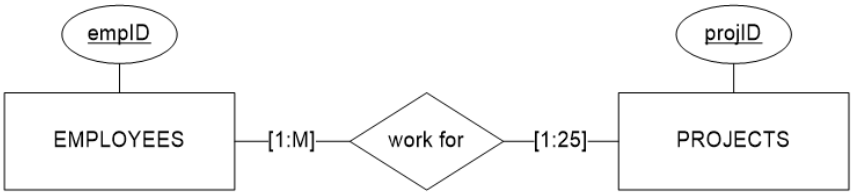



*Figure 5: The work_on relationship*


*Figure 6: Employees, Projects and Clients*

and grouping allow us to count specific aspects of the set (or specific members). This determines what is counted, and hence the kind of cardinality constraint being examined. We examine both these operators in more detail.

Selection ($\sigma$) may be performed on any attribute combination within one or more relations. We decompose this further into: $\sigma$ performed on an identifying attribute of a participating entity (or entity combination), denoted by $\sigma^{\text{ID}}$, and a $\sigma$ performed on a non-identifying attribute $\sigma^{\text{A}}$. We separate the two because there is a direct correspondence between an identifier and an entity instance, which in turn allows us to ask the question "how many" (i.e., count) as applicable to a given entity (e.g., employee). While there has been some deliberation in literature about the distinction between entities and attributes, we do not get into that discussion and instead refer the reader to previous work (Weber 1996; Kop and Mayr 1998), while assuming that the distinction is beneficial. Related to the use of $\sigma^{\text{ID}}$, we know from the query languages that a $\sigma$ need not contain only a single identifier value, and could instead be evaluated over a set of identifiers using either multiple `OR` clauses, (in this case we can combine the set of identifiers using an `IN` operator as well). We denote a set of identifiers by: $\sigma^{\text{ID-Set}}$. This leads to the classification of a conceptual model constraint category not previously discussed in literature, that of *set-level constraints* (e.g., *There should be no more than 5 projects associated across the set of employees: 'E004', 'E005', 'E007'*).

Returning to $\sigma^{\text{A}}$, when the attribute A is from a participating entity class, the $\sigma^{\text{A}}$ serves to establish a predicate on the relevant entity class. This does not impact the nature of the constraint, just the scope of the set membership on which it is defined. For example, consider constraints:
C1a: *An employee can work for between 1 and 25 projects;* and C1b: *An employee with the designation of "Senior Consultant" can work for between 5 and 10 projects* (this version also considers a *single employee*, but has an additional $\sigma$ condition to only consider an employee if they are a senior consultant).

The fundamental form of both these constraints is the same in that they capture the number of members of a "counted" entity class (projects) that co-occur with each member of a "fixed" class (employees). The only difference between them is a predicate that restricts the membership of the "fixed" set (or the "counted" set). This principle holds whether we are considering instance-level or set-level constraints. As an example, consider the following constraints:
C2a: *There should be no more than 50 projects associated with the set of all employees;* and C2b: *There should be no more than 5 projects associated with the set of all employees with a security clearance of "alpha"* (this version also considers a *set of employees*, but has an additional

$\sigma$ condition to only consider those employees with a security clearance of "alpha").

Both these constraints are set-level constraints where the "fixed" class is employees, and the cardinality limits the association with projects (the "counted" class). To summarize, using predicates provides flexibility in expressing a constraint type, but does not change the nature of what is counted.

The case may be made for separate consideration of attributes in $\sigma^{A}$ that are mutual properties of entities in a relationship, e.g., counting employees who received a particular level of feedback or rating for their work on the project. We adopt a philosophy previously presented in literature suggesting that relationships should not have attributes of their own, but these must be either modeled via a separate relationship or attaching an additional associated (usually, weak) entity class (Wand et al. 1999). The function of these attributes in constraints consequently reduces to that of predicates. Even if we do not adopt this view, the only difference is that we end up with a generalized version of constraint types that allow for counting of not only entity instances but also attribute values within a relationship. Either way, the target relational constraint can be captured using conceptual model constraints.

The grouping operation $\Im$ is a convenient extension of the $\sigma$ applied to specific entity instances. Instead of checking the count for a single instance, it does so for each distinguishable instance in a relationship. For example, we could perform a count of projects for a single employee by specifying an employeeID with the $\sigma$ operator. Using $\Im$ allows us to conveniently perform a similar count for each employee without having to manually enumerate each employee's identifier. When a $\sigma$ is used in conjunction with a $\Im$, it (the $\sigma$) serves the role of either restricting the scope of the $\Im$ (if the attributes the $\sigma$ and $\Im$ are applied on are identical), or that of predicate filtering. As is the case for $\sigma$, we consider $\Im$ on identifying attributes without loss of expressiveness.

We present our analysis of the semantics of cardinality in the next two sections. The basic approach followed is to examine combinations of projection, selection and grouping and their correspondence to various cardinality constraints. For consistency, we assume the underlying conceptual schema is developed using the ER model (Appendix A summarizes the syntax used), and follow cardinality terminology developed by Liddle (Liddle et al. 1993) and extended by us (Currim and Ram 2012). These papers also contain a resolution among constraint naming conventions used in a variety of semantic models and previous frameworks. To formally clarify the semantics of each constraint type, we use first order logic and integrate it with a sample annotation syntax (recapped in Appendix B using BNF). A (simplified) corporate schema[6] used to illustrate the constraint semantics follows.

## 4.1 CONSTRAINTS APPLICABLE TO ENTITY CLASSES

A **class** constraint restricts the number of members in the entity class. An **attribute** constraint restricts the cardinality of the number of values an entity instance can have (for an attribute), e.g., we may wish to restrict how many cities a project can span. A **domain** cardinality constraint (encountered and formally defined by us during the process of establishing completeness relative to relational algebra) restricts the number of possible values an attribute can take on for the set of entity instances. The domain cardinality does not restrict the number of cities for a *single* project entity, rather specify a restriction *across all* (or some subset of) projects.

The cardinality of a finite set $E$ (e.g., entity class) is written as $|E|$. Typically, the constraint is specified with a lower and upper bound. We represent the range of values between the bounds by the set *Card*. Since $E$ is finite, we say $|E| \in Card$, where $Card \subseteq \mathbb{N} \cup \{\mathtt{M}\}$; $\mathbb{N}$ is the set of natural numbers and the symbol $\mathtt{M}$ denotes "many" (unrestricted upper bound). For convenience, in

[6] To preserve the ternary semantics of cardinality constraints (as these are easier to read and interpret), we don't model the change in cardinality due to the introduction of a weak association class and the consequent change in degree of the relationship to quaternary.

the syntactical version, we specify the lower and upper limits of the constraint as ⟨min⟩ and ⟨max⟩, where ⟨min⟩ ⊆ ℕ, and ⟨max⟩ ⊆ ℕ ∪ {M}. We use *A* for attributes, and *P* to represent a predicate. $P(e(A_{km}))$ signifies the $m^{th}$ predicate defined on attribute $A_k$. For simplicity, we only describe the use of predicates for entity class cardinality and do not repeat the syntax for the constraint bounds (i.e., "IN [⟨min⟩:⟨max⟩]") for later constraints. For an entity class *E*, we use $\pi^{A}(E)$ to denote the projection of the values of attribute A across all members of *E*, while $\pi^{A}(e)$ represents the projection for a specific entity *e*, where $e \in E$.

In Table 2 we consider combinations of relational algebra operators as they apply to class cardinality constraints. In Table 3, we use $\pi^{cA}$ operations instead of $\pi^{c*}$, which leads to attribute and domain cardinality constraints. In the first column we show the relational algebra expression, followed by the semantics of the specified constraint (natural language). The final column has the equivalent SQL query.

In the next sub-section, we discuss constraints applicable to relationships.

## 4.2 CONSTRAINTS APPLICABLE TO RELATIONSHIPS

An *interaction relationship* relates members of one entity class to members of one or more entity classes. Interaction relationship constraints are classified into participation, set-participation, projection, co-occurrence, set-co-occurrence, appearance, set-appearance, appearance-across-R, and set-appearance-across-R constraints (the latter four types being applicable for unary relationships). We begin by briefly describing each type of constraint and illustrating its semantics using the relationship assign from Figure 7. Thereafter, each is discussed in detail. To formally clarify the semantics of each constraint type, we use first order logic.

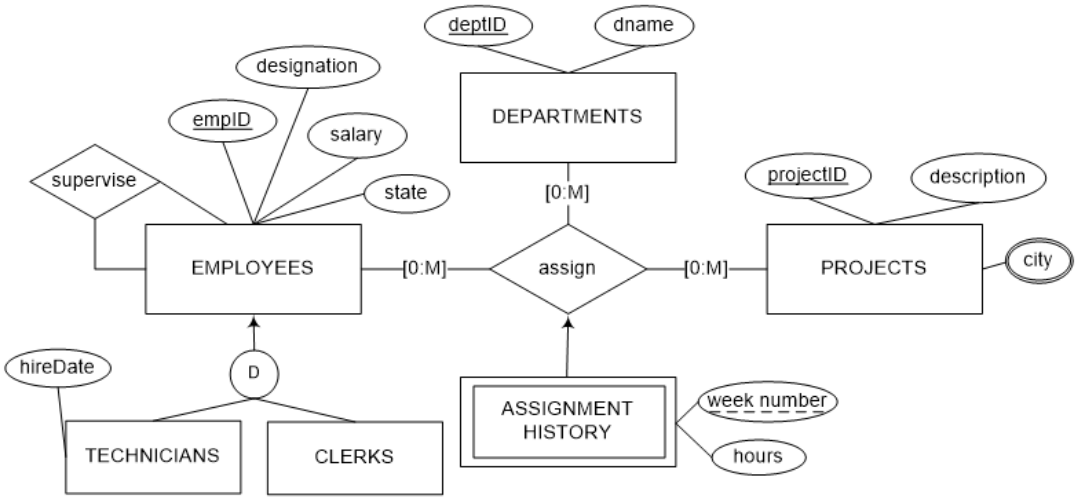
**Participation** constraints look at a relationship (e.g., employees being *assigned* to projects by departments), and ask the question, "How many times can an entity (e.g., an employee) participate in the relationship?" **Set-Participation** constraints (newly introduced by us) look at a (sub)set of entities (e.g., employees belonging to the states of California, Arizona and New Mexico) and ask, "How many times can the set of entities considered together participate in the relationship?" The generalized version of both these rule types consider not just a single entity class, but also entity combinations, for example "How many times can a given combination of employee and department be assigned in the assign relationship?"

For the formal definition, we introduce *entity from relationship* projection. Assume a relationship *R* (e.g., the assign relationship) is formed by the entity classes $E_1, \ldots, E_i$ and each element $r \in R$ (e.g., a specific assignment instance). Then we define $\pi^{E_i}(r)$ as the projection of the entity from class $E_i$ belonging to the relationship instance *r*, while $\pi^{E_i}(R)$ projects all members from $E_i$, present within the relationship *R*. For the set-participation constraints, to define a subset $C_i$ of an entity class $E_i$, we use $C_i \subseteq E_i$.

**Projection** constraints look at the relationship and restrict how many distinct entity instances can occur across the set of relationship instances. Thus, "How many different projects can there be in all (in the assign relationship)?" is an example of the projection constraint. The generalized version of this constraint examines entity combinations from multiple entity classes.

**Co-occurrence** constraints consider an entity already known to be participating in a relationship, and ask how many members of another entity class can co-occur with it. Thus, for example, one could ask, "Given a project that exists in the assign relationship, how many distinct departments can co-occur with it?" The *Set-Co-occurrence* constraint (newly introduced by us) looks at a set of entities known to be participating in a relationship and asks how many members of another entity class can co-occur with it. For example, "For projects classified as high-security, how many different meetings co-occur with them?" The generalized version of this constraint considers entity combinations.

So far, we have only examined cardinality constraints defined over instances of a relationship.




```
EMPLOYEES (empID, designation, state, salary)
DEPARTMENTS (deptID, dname)
MANAGERS (empID, hireDate)
PROJECTS (projectID, description)
PROJECT_CITIES (projectID, city)
SUPERVISE (empID, supervisorID)
ASSIGN (projectID, empID, deptID, weekNumber, hours)
```

*Figure 7: A simplified corporate schema (ER and Relational versions)*

*Table 1: Semantics and sample syntax for Class and Attribute cardinality constraints*

| | |
|---|---|
| **Class Cardinality** | `CARD-C` $(E)$ `IN` [⟨`min`⟩:⟨`max`⟩] *is defined as:* $\lvert\{ e : e \in E \}\rvert \in Card$ |
| ***(with predicates)*** | `CARD-C` $(E$ [⟨`predicate conditions`⟩]) $\lvert\{ e : e \in E \wedge P\ (e(A_{11}), e(A_{12}), \ldots, e(A_{k1}), \ldots, e(A_{km}))\ \}\rvert \in Card$ |
| **Attribute Cardinality** | `CARD-A` $(E, A)\ \forall\ e \in E, \lvert\{ \pi^{A}(e) \}\rvert \in Card$ |
| **Domain Cardinality** | `CARD-D` $(E, A) \lvert\{ \pi^{A}(E) \}\rvert \in Card$ |

Including the notion of counting values across attributes rather than tuples (i.e., the participating entity classes, rather than the relationship instances) leads us to the examination of forms of the *appearance constraint*. These are applicable for unary relationships. For example, we may ask, "*How many roles supervisee, supervisor can a single employee play within a single relationship instance of the* supervise *relationship?*" Thus, if an employee can supervise herself, then she can play two roles. This is an example of the **appearance** constraint, which restricts the number of roles in which a given member $e$ of an entity class $E$ can appear in any instance $r$, $r \in R$. It applies to an interaction relationship $R$ in which the same underlying entity class $E$ participates in $R$ in different roles $L_1, \ldots\ L_k$. The **Set-Appearance** constraint restricts the number of roles a set of entities can play in any single entity instance. An **Appearance Across-R** constraint restricts the number of roles in which a given member of $E$ can appear across all instances of $R$ (or some subset $R$', $R' \subseteq R$). A

*Table 2: Semantic Analysis of Combinations of SQL Operations for Entity Classes*

| **Algebraic Operator** | **Semantics for an Entity Class** | **SQL Equivalent** |
|---|---|---|
| $\pi^{c^*}$ i.e., applied on the entity (tuple) | *Class cardinality* constraint (number of members in an entity class) | `SELECT count(*) FROM employees;` |
| $\pi^{c^*}$, $\sigma$ | *Class cardinality* with predicate | `SELECT count(*) FROM employees WHERE state='CA';` |
| $\pi^{c^*}$, $\Im$ | *Class cardinality* of an attribute-defined subclass/subset (while the equivalent cardinality may be computed by using state as a discriminator to define subclasses—we feel that it is somewhat artificial to require creation of subclasses simply to enforce cardinality). *Note:* it is not meaningful to use the identifier as the grouping attribute—since that will always yield one group per entity (i.e., any further count will always equal to 1) | `SELECT count(*) FROM employees GROUP BY state;` |
| $\pi^{c}$, $\Im$, $\sigma$ | *Class cardinality* of a subset/subclass (based on predicate) | `SELECT count(*) FROM employees WHERE salary > n GROUP BY state;` |
| $\pi^{c}$, - | For completeness, we illustrate the effect of combining queries using the `MINUS` operator. As can be seen, it is not meaningful to perform such counts since it will return the count of the first query unless the cardinalities of the two sets in question are identical, whereupon it will return no results. Instead, the set difference should be performed first, and a single count taken on the resulting set (which does not change the nature of what is counted). | `SELECT count(*) FROM employees MINUS SELECT count(*) FROM managers;` |

**Set-Appearance Across-R** constraint restricts the number of roles in which a given set $C$, $C \subseteq E$, of entities can appear across $R$ (or some subset $R$', $R' \subseteq R$). We discuss the formal semantics and syntax in Table 7. For convenience, we define role projection operations. Role projection $\pi^{L}(r, e)$ is defined as: $R \times E \rightarrow \boldsymbol{P}(L)$, where $\boldsymbol{P}(L)$ is the power-set of $L$. It takes as input a relationship instance $r$, an entity instance $e$, and returns the set of roles that entity instance plays in the instance r. We generalize this to allow for $\pi^{L}(r, C)$, where $C \subseteq E$, and define it as: $R \times \boldsymbol{P}(C) \rightarrow \boldsymbol{P}(L)$. Similarly, we define operations to allow for projection of roles across instances of $R$, both: $\pi^{L}(R, e)$ and $\pi^{L}(R, C)$. Doing so allows us to keep the definition of the constraints compact.

Now that we have described the formal semantics for the various relationship constraint types, we consider combinations of relational algebra operators and SQL as they apply to the relevant cardinality constraints.

For appearance constraints, the query is non-

*Table 3: Semantic Analysis of Combinations of SQL Operations for Attributes and Domains*

| Operator | Semantics for an Entity Class | SQL Equivalent |
|---|---|---|
| $\pi^{cA}$ i.e., applied on a single attribute | *Domain cardinality. Note*: it is not useful to count the identifying attribute—since that will always be equal to the cardinality of the class itself | `SELECT count(distinct designation) FROM employees;` |
| $\pi^{cA}$, $\sigma$ | *Domain cardinality* with a predicate (if non-identifying attribute in the selection $\sigma$) | `SELECT count(distinct designation) FROM employees WHERE state='CA';` |
| $\pi^{cA}$, $\sigma^{ID}$ | *Attribute cardinality* for a specific entity instance (if part of the identifying attribute is included in the count). This is only meaningful for a multi-valued attribute (which in relational terms would be translated into a separate table) | `SELECT count(distinct city) FROM project_cities WHERE projectID='J15';` |
| $\pi^{cA}$, $\Im$ | *Domain cardinality* within each defined subset | `SELECT count(distinct designation) FROM employees GROUP BY state;` |
| $\pi^{cA}$, $\Im$, $\sigma$ | *Domain cardinality* within each defined subset (with a predicate) | `SELECT count(distinct designation) FROM employees WHERE salary > n GROUP BY state;` |
| $\pi^{cA}$, - | As discussed for the $\pi^{c*}$ case, it is not meaningful to perform counts in this manner. | `SELECT count(distinct state) FROM employees MINUS SELECT count(distinct state) FROM managers;` |

*Table 4: Semantics and syntax for Participation (instance and set-level) constraints*

| | |
|---|---|
| **Participation** (generalized) | `CARD-R-PT` $(R, E_1, \ldots, E_i)\ \forall\ (e_1 \in E_1, \ldots, e_i \in E_i), \lvert\{r : r \in R \wedge \pi^{E_1}(r) = e_1 \wedge \ldots \wedge \pi^{E_i}(r) = e_i\}\rvert \in Card$ |
| **Set Participation** | `CARD-R-PT-SET` $(R, E_1, \ldots, E_i)\ \lvert\{r : r \in R \wedge \pi^{E_1}(r) \in C_1 \wedge \ldots \wedge \pi^{E_i}(r) \in C_i\}\rvert \in Card$ |

*Table 5: Semantics and sample syntax for Projection constraints*

| | |
|---|---|
| **Projection** | `CARD-R-PJ` $(R, E_1, \ldots, E_i)\ \lvert\{r : r \in \pi^{E_1,\ldots,E_i}(R)\}\rvert \in Card$ |

trivial since SQL does not have an in-built function to project roles. Unlike the extended projection operators defined in Table 7 (which allow for working with power sets), we further augment the

*Table 6: Semantics and sample syntax for Co-occurrence (instance and set-level) constraints*

| | |
|---|---|
| **Co-occurrence** | CARD-R-CO $(R, (E_1, \ldots, E_i), (E_{i+1}, \ldots, E_j)) \; \forall \; s \in \pi^{E_1,\ldots,E_i}(R), \; \lvert\{t : t \, \pi^{E_{i+1},\ldots,E_j}(R) \wedge (s \circ t \in R)\}\rvert \in Card$ |
| **Set Co-occurrence** | CARD-R-CO-SET $(R, (E_1, \ldots, E_i), (E_{i+1}, \ldots, E_j)) \; \lvert\{t : t \in \pi^{E_{i+1},\ldots,E_j} \wedge s \in \pi^{E_1,\ldots,E_i} \wedge (s \circ t \in R) \wedge \pi^{E_1}(s) \in C_1 \wedge .... \wedge \pi^{E_i}(r) \in C_i\rvert \in Card$ |

*Table 7: Semantics and sample syntax for various kinds of Appearance constraints*

| | |
|---|---|
| **Appearance** | CARD-R-AP $(R, E, L_1, \ldots, L_i) \; \forall \; e \in E, \lvert \{ l : l \in E \, \pi^L(r, e) \}\rvert \in Card$ |
| **Set Appearance Across-R** | CARD-R-AP-SET $(R, E, L_1, \ldots, L_i)\lvert\{ l : l \in \pi^L(r, C) \wedge C \subseteq E \}\rvert \in Card$ |
| **Appearance Across-R** | CARD-R-APAR $(R, E, L_1, \ldots, L_i) \; \forall \; e \in E, \lvert\{ l : l \in E \, \pi^L(R, e) \}\rvert \in Card$ |
| **Set Appearance Across-R** | CARD-R-APAR-SET $(R, E, L_1, \ldots, L_i)\lvert\{ l : l \in \pi^L(R, C) \wedge C \subseteq E \}\rvert \in Card$ |

*Table 8: Using only π*

| **Algebraic Operator** | **Semantics for an Entity Class** | **SQL Equivalent** |
|---|---|---|
| $\pi^{c*}$ | *Projection cardinality*, i.e., number of association instances in the relationship. | `SELECT count(empID) + count(supervisorID) FROM assign;` |
| $\pi^{cA}$ | *Projection cardinality* restricted to a participating entity class. | `SELECT count(distinct empID) FROM assign;` |
| $\pi^{cA,B}$ | *Projection cardinality* restricted to a combination of entities from participating entity classes. We don't consider concatenation (or correspondingly: adding multiple attributes in the $\mathfrak{I}$ clause) in further examples to preserve simplicity, but note that it allows for combinations of entities from participating classes to be considered rather than just from one class. | `SELECT count(distinct concat(empID,deptID) ) FROM assign;` |

functions in the presence of multiple roles (see Table 11 for details). This also demonstrates the advantage of using a semantics-based approach, rather than re-specifying the programming logic for each instance of the constraint. For simplicity, we assume a role-projection function exists for relational algebra and demonstrate one set of feasible solutions assuming two possible roles (based on the supervise relationship).

In this section we discussed the equivalence of conceptual modeling constraints and combinations of relational algebra operations. In doing so, we introduce new constraints at the set level. A similar exercise can be carried out for other modeling constructs (e.g., superclasses and subclasses). However, we feel the relational algebra mappings for entity classes, attributes and interaction relationships sufficiently demon-

*Table 9: Using $\pi^{c*}$, $\sigma$ and $\Im$*

| Operator | Semantics for an ER Relationship | SQL Equivalent |
|---|---|---|
| $\pi^{c*}$, $\sigma^{ID}$ | *Participation cardinality*, i.e., number times an entity (or entity combination) participates in a relationship. | `SELECT count(*) FROM assign WHERE projectID='P01';` |
| $\pi^{c*}$, $\Im^{ID}$ | *Participation cardinality* (for every projectID in the relationship). | `SELECT count(*) FROM assign GROUP BY projectID;` |
| $\pi^{c*}$, $\sigma^{A}$ | *Projection cardinality*, i.e., number of association instances in the relationship matching the property specified in the $\sigma$. | `SELECT count(*) FROM assign WHERE hours = n;` |
| $\pi^{c*}$, $\Im^{ID}$, $\sigma^{A}$ or $\pi^{c*}$, $\Im^{ID\text{-}x}$, $\sigma^{ID\text{-}y}$ | *Participation cardinality*. The combination of $\sigma$ and $\Im$ (performed on the same attribute) simply restricts which tuples of the entire set are split into groups. A $\Im$ performed on a different attribute than the $\sigma$ leads to a participation cardinality check with the $\sigma$ performing the role of predicate filtering. It remains a form of participation cardinality. We use $\Im^{ID\text{-}x}$, $\sigma^{ID\text{-}y}$ to indicate that the grouping and selection are on different identifying attributes (i.e., different entity classes). The $\sigma^{ID\text{-}y}$ condition may involve a single value or a set of values. The accompanying SQL example uses a set of values. A combination of $\sigma^{A}$ with $\sigma^{ID\text{-}y}$ may also be done. A similar pattern is observed for co-occurrence constraints in Table 10 (discussion not repeated in the interests of brevity). | `SELECT count(*) FROM assign WHERE hours > n GROUP BY projectID; SELECT count(*) FROM assign WHERE empID IN ('E04', 'E05', 'E07') GROUP BY projectID;` |
| $\pi^{c*}$, $\sigma^{ID\text{-}Set}$ | *Set-Participation cardinality*, i.e., how often do a set of entities considered together participate in a relationship. | `SELECT count(*) FROM assign WHERE projectID IN ('P01','P02','P03');` |

strates the important aspects of our approach.

## 5 Evaluation and Testing

To complement our approach for showing completeness of a cardinality taxonomy, we decided to develop a prototype system to serve as a proof-of-concept for the framework. To properly automate the development of the constraint translation module, we realized that the system needed to be aware of the relational schema (to reference column and table names in the SQL and triggers, for example). While our over-arching purpose was to evaluate whether the SQL mapping logic from our conceptual constraint specifications was correct, we felt the experience would additionally inform us of any database-implementation issues that could arise for the translated constraints. This motivated our development of the CARD (Constraint Automated Representation for DBMSs) system (Figure 8).

*Table 10: Using $\pi^{cA}$, $\sigma$ and $\mathfrak{I}$*

| Operator | Semantics for an ER Relationship | SQL Equivalent |
|---|---|---|
| $\pi^{cA}$, $\sigma^{ID}$ | *Co-occurrence cardinality*, i.e., for an entity defined in the $\sigma$ clause, how many distinct entity instances co-occur with it ($\pi$ clause)? | `SELECT count(distinct empID) FROM assign WHERE projectID='P01';` |
| $\pi^{cA}$, $\mathfrak{I}^{ID}$ | *Co-occurrence cardinality* for each entity defined by the grouping attribute(s). | `SELECT count(distinct empID) FROM assign GROUP BY projectID;` |
| $\pi^{cA}$, $\sigma^{A}$ | *Projection cardinality*, restricted to a subset of association entities (matching the property specified in the $\sigma$) from the participating entity class (specified by the $\pi^{cA}$). | `SELECT count(distinct empID) FROM assign WHERE hours > n;` |
| $\pi^{cA}$, $\mathfrak{I}^{ID}$, $\sigma^{A}$ or $\pi^{c*}$, $\mathfrak{I}^{ID\text{-}x}$, $\sigma^{ID\text{-}y}$ | *Co-occurrence cardinality*. When used in combination, the grouping attribute(s) end up being the fixed aspect of the co-occurrence cardinality (it does not matter that a subset of entities is specified by the $\sigma$, since these are broken up for individual consideration by the $\mathfrak{I}$). The projection (counted) attribute plays its usual role, while the attribute in the $\sigma$ clause works to filter the set of rows under consideration similar to a predicate. | `SELECT count (distinct empID) FROM assign WHERE projectID IN ('P01', 'P02', 'P03') GROUP BY deptID, projectID;` |
| $\pi^{cA}$, $\sigma^{A}$ or $\pi^{cA}$, $\sigma^{ID\text{-}Set}$ | *Set co-occurrence cardinality*, i.e., for a given set of entities (fixed by $\sigma$) how many instances of another entity class are associated with them (counted in $\pi$). | `SELECT count(distinct empID) FROM assign WHERE projectID IN ('P01', P02');` |

Users interact with the system and choose one of the options to input a schema and associated constraints. The system then uses knowledge of standard ER-to-relational conversion logic to develop the SQL (DDL statements) for table creation. In addition, it takes advantage of the knowledge of the schema and the cardinality-to-SQL mapping we described in our paper, to generate the associated triggers. Currently, freeware and commercial tools exist that can do the first part (i.e., take a conceptual schema developed using ER modeling or UML and convert it into relations with a limited number of structural constraints). The latter part is new, and developed as an extension by us to demonstrate the practical feasibility of the translation based on our completeness discussion. A system like CARD can serve the software development lifecycle (SDLC) by generating relational triggers to manage the advanced constraints, which has the two-fold advantage of improving both database integrity and programmer productivity.

The prototype (currently implementing a subset of the cardinality constraints) has been developed in Java, and is available over the Internet (Currim et al. 2010). For our prototype we picked Oracle as the target DBMS, since it is widely-used. While the SQL table creation code is designed to be ANSI compliant and work across platforms, the constraint triggers are generated in PL/SQL and are Oracle specific (however, the core logic can

*Table 11: Limiting roles (for Appearance constraint variants)*

| Operator | Semantics for an ER Relationship | SQL Equivalent |
|---|---|---|
| $\pi^{L(r,e)}$ | *Appearance cardinality*, i.e., restricts how many roles a given entity can play in a single relationship instance. *Note*: while we use the Oracle specific `DUAL` system table, this can be translated into another platform like SQL Server either by removing the reference to `DUAL` (or for DB2: by using the `SYSIBM.SYSDUMMY1` instead) or creating a schema specific table that simply contained a single row/column to use instead of dual. | `SELECT COUNT(*) FROM (SELECT 'One Role' FROM dual WHERE EXISTS (SELECT * FROM supervise WHERE (empid='E01' OR superviseID='E01') ) UNION SELECT 'Two Roles' FROM dual WHERE EXISTS ( SELECT * FROM supervise WHERE empid=superviseID AND empid='E01') );` |
| $\pi^{L(R,e)}$ | *Appearance Across-R cardinality* (new). This constraint restricts the number of roles a single entity can play across all relationship instances. In this case, a `UNION ALL` (bag semantics) is required to prevent loss of information when the same entity (identifier) is selected from two different roles. | `SELECT COUNT(*) FROM (SELECT distinct empID FROM supervise WHERE empid = 'E01' UNION ALL SELECT distinct supervisorID FROM supervise WHERE supervisorID = 'E01');` |
| $\pi^{L(r,C)}$ | *Set Appearance cardinality* (new). This constraint restricts the number of roles a set of entities can play in a single relationship instance. | `SELECT COUNT(*) FROM (SELECT 'One Role' FROM dual WHERE EXISTS (SELECT * FROM supervise WHERE (empid IN ('E01', 'E02') OR superviseID IN('E01', 'E02')) ) UNION ALL SELECT 'Two Roles' FROM dual WHERE EXISTS (SELECT * FROM supervise WHERE (empid IN ('E01', 'E02') AND superviseID IN('E01', 'E02')) );` |
| $\pi^{L(R,C)}$ | *Set Appearance Across-R cardinality* (new). This is the set equivalent of the appearance across-R constraint and restricts how many roles a set of entities can appear in, across all relationship instances. | `SELECT COUNT(*) FROM (SELECT 'Role Employee' FROM dual WHERE EXISTS (SELECT * FROM supervise WHERE empid IN ('E01', 'E02') ) UNION ALL SELECT 'Role Supervisor' FROM dual WHERE EXISTS (SELECT * FROM supervise WHERE superviseID IN ('E01', 'E02') ));` |

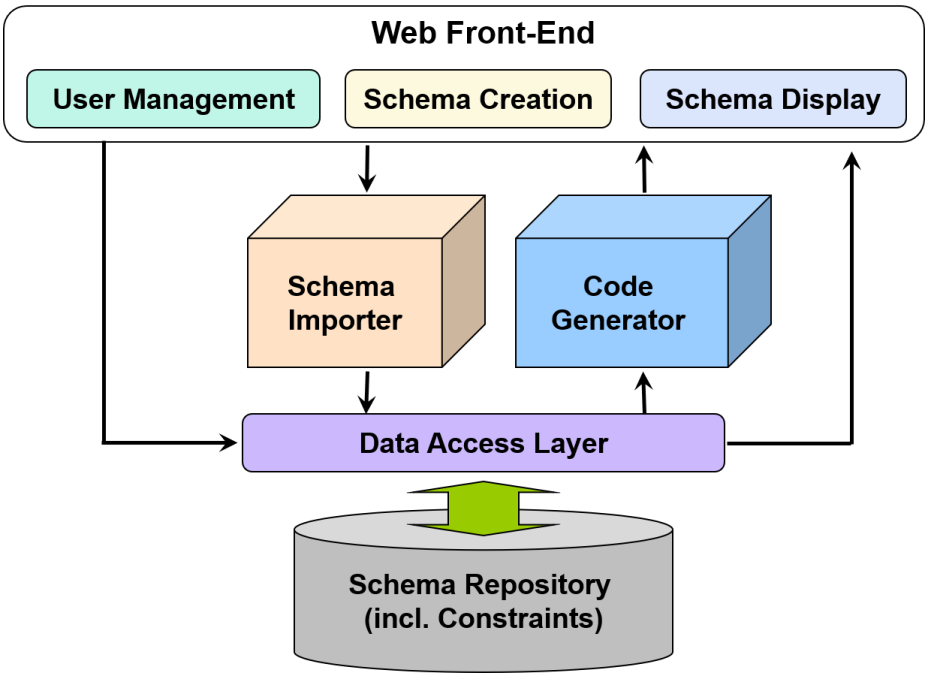



*Figure 8: Architecture of the CARD system*

*Table 12: Implication of constraint parameters for triggering events*

| **Constraint Property** | **Trigger Fired On** |
|---|---|
| Minimum cardinality specification of > 0 | `DELETE` |
| Maximum cardinality of < `M(any)` | `INSERT` |
| Predicates on Entity Class involved in Constraint | `UPDATE` |

be modified in a straightforward manner for other platforms). The current interface supports creating schemas directly via a guided specification interface, and importing schemas developed either in Visio's XML drawing format or directly in a canonical ER-XML format developed by us (an XML Schema specification that allows a standard representation of an ER schema). This can be extended in the future to allow additional XML representations for ER or UML (since the core constructs and their purpose are similar). All options allow users to specify entity classes (including strong and weak classes), relationships (interaction, inclusion, etc.), and constraints.

We briefly describe the trigger generation strategy employed. Depending on the nature of the constraint summarized in Table 12, the firing event is set to `INSERT`, `DELETE` or `UPDATE`. Since the integrity constraints must be checked prior to any possible violations, we specify it be checked `BEFORE` the database event takes place. A minimum cardinality specification of > 0, implies the need for a trigger fired on deletions, while a specified maximum cardinality (other than simply "many") requires checking counts before tuple insertion. We assume identifier values are unchangeable. If the database allows updates to primary key values, then update event triggers must be used for both of the previous cases as well. Triggers checked on update are also needed when a predicate is specified for the constraint. Our current system assumes that all constraint vi-

```
CREATE OR REPLACE TRIGGER ⟨Trigger Name⟩
   BEFORE ⟨Firing Event as described in Table 12⟩
   ON ⟨Affected Relation⟩
   DECLARE
     ⟨Variables including Cardinality Limits⟩
   BEGIN
              ...
     ⟨Perform count based on SQL clause as described in section 4⟩
     ...
     IF ⟨Cardinality Limits are Violated⟩ THEN
       ⟨Perform Database Actions to Manage Violation⟩
     END IF;
     ...
   END;
/
```

*Figure 9: Generalized Trigger Template (Oracle)*

olations must be prevented (hence an application error is raised to block the database operation). However, the user may choose to specify different actions that could be taken, including simply generating warnings, logging the violations, and so on (discussed in depth in active database literature (Paton and Díaz 1999). We leave the handling of violationaction refinements for future work.

Figure 9 shows a generalized trigger template. While the automated generation may seem straightforward, we encounter locking granularity issues for multi-user environments. Typically, a transaction that only inserts a new tuple would not obtain an exclusive table-level lock (since it is overly restrictive). This can cause constraint violations. Using our earlier example of, "any supplier and project combination can appear in the SubmitBids relationship between [$h$ .. $k$] times", let's assume we were trying to enforce the upper bound of $k$ where currently there were $k$-1 entries for an $\langle s, j\rangle$ pair. Suppose two transactions $t1$ and $t2$ both attempt to insert a new instance of $\langle s, j\rangle$ with some $p1$, $p2$, where $p1 \neq p2$, and the associated triples were new to the relation (i.e., both ⟨s, j, p1 ⟩ and $\langle s, j, p2\rangle \notin$ SubmitBids). Given that each would not see the uncommitted inserts of the other transaction, they would both count $k$-1 entries, and permit the insertion to proceed (since the insertion likely would not violate any other database constraints), resulting in $k$+1 instances of the $\langle s, p\rangle$ pair (a violation). As can be seen, this requires the use of an exclusive lock on affected table which leads to inefficiencies.

For special classes of constraints, notably those applied at the set level, another approach is to maintain a constraint metadata table (CMT). The CMT would contain the type, cardinality limits, and current counts for the constraint in question. Instead of performing the count on the data table (e.g., SubmitBids) each time an insert took place, the trigger could look up the CMT (the tuple corresponding to the constraint in question would be locked) and suitably adjust the attribute for the current count. The problem with applying this to instance-level constraints (that are not restricted to a small set of entities), is lack of scalability. There could be a large number of suppliers (for example) and we would not wish to store, one entry in the CMT per supplier. Pre-sorting or an index variant (with a supplier-count pair) could make this

option more efficient at an instance level, but we feel that an effective and generalizable solution would require native DBMS support for cardinality constraint enforcement. Further, while some would argue for application level support instead, we feel instead that it would be more efficient to perform the operations at the database level (rather than transmit the data to the application to do the check). In the case of the two transactions *t1* and *t2* described previously, native DBMS support could permit better scheduling of potentially conflicting transactions and re-use of knowledge from recently performed counts.

## 6 Conclusions and Contributions

An information system may need to manage a large number of rules from governmental, industry and organizational requirements. Rules assert business structure, can influence organizational behavior (Hay et al. 2000) and describe a state of affairs that the business wants to exist (Morgan 2002). Understanding rules, including cardinality, and modeling them in the conceptual schema in a visible manner is crucial, because if they are missed at this stage, they may never be enforced (or be applied inconsistently) when the database is implemented (Shao and Pound 1999). In addition to the value of establishing completeness of a rule framework from a research perspective, we also see utility from the perspective of model-driven architecture (MDA) (OMG 2001). The connection between conceptual modeling and MDA is important in information systems development (Kop et al. 2007), and approaches like MDA benefit from standardized constraint representation (J. B. Warmer and A. G. Kleppe 2003). Without classification, there are limitations to enabling a standardized syntax to represent constraint types and the subsequent mapping into code. Establishing completeness likewise benefits CASE tools incorporating such frameworks.

We have discussed an approach to establishing completeness of cardinality constraint frameworks. In doing so, we added a constraint category (set-level), orthogonal to existing types of cardinality rules not previously seen in literature. Modeled constraints may be represented in the conceptual schema by the analyst using a variety of syntactical approaches for ER or UML. Subsequently, a well-defined mapping can be used to come up with the corresponding implementation code.

The basic approach to supporting constraint implementation is by translating the specific constraint logic into SQL and embedding that within triggers or procedures. Given an ER schema and the associated constraints, the relevant count of data values can be checked against the constraint limits. We described a prototype currently under development to automatically translate modeled constraints from the conceptual design level into database triggers. Such checks can be implemented using Oracle's PL/SQL or SQL Server's Transact-SQL. Our efforts also lead us to recommend better DBMS support for cardinality to improve the efficiency of the constraint checking triggers.

We feel our approach shows the value of concise semantic-based rule annotation schemes.

Establishing completeness supports constraint incorporation into CASE tools for productivity gains during the development lifecycle. The platform independent conceptual taxonomies (which can be represented using a variety of syntaxes including XML or the annotation scheme shown in Appendix B) support distributed development while fitting well into the service-oriented computing paradigm. Although we used the ER model in describing constraints in this paper, the extension to UML and other modeling languages is straightforward. We feel future research directions could test whether similar approaches to completeness can be extended to other kinds of database rules, including temporal and spatial integrity constraints.

## References

Atzeni P., Chen P. P. (1981) Completeness of query languages for the entity-relationship model.

In: Proceedings of the Second International Conference on the Entity-Relationship Approach to Information Modeling and Analysis. North-Holland Publishing Co., pp. 109–122

Bajaj A., Ram S. (2002) SEAM: A State Activity Entity Model for a Well Developed Workflow Development Methodology. In: 14 (2), pp. 415–431

Campbell D. M., Embley D. W., Czejdo B. D. (1985) A relationally complete query language for an entity-relationship model. In: Proceedings of the Fourth International Conference on Entity-Relationship Approach. IEEE Computer Society, pp. 90–97

Chen P. P.-S. (1976) The entity-relationship model—toward a unified view of data. In: ACM Transactions on Database Systems (TODS) 1(1), pp. 9–36

Codd E. F. (1972) Relational completeness of data base sublanguages. IBM Corporation

Currim F. (2004) Spatio-Temporal Set-Based Constraints In Conceptual Modeling: A Theoretical Framework and Evaluation. PhD thesis, University of Arizona, Tucson, AZ

Currim F., Neidig N., Kampoowale A., Mhatre G. (2010) The CARD System. In: Proceedings of the Twenty-ninth International Conference on Entity-Relationship Approach. Springer, pp. 433–437

Currim F., Ram S. (2012) Modeling spatial and temporal set-based constraints during conceptual database design. In: Information Systems Research 23 (1), pp. 109–128

Date C. J. (2000a) An introduction to database systems. In: Boston: Pearson/Addison Wesley 27(983), p. 22

Date C. J. (2000b) What not how: the business rules approach to application development. Addison-Wesley Professional

Ferg S. (1991) Cardinality Constraints in Entity-Relationship Modeling.. In: ER, pp. 1–30

Halpin T., Morgan T. (2010) Information modeling and relational databases. Morgan Kaufmann

Hay D., Healy K. A., Hall J., Bachman C., Breal J., Funk J., Healy J., McBride D., McKee R., Moriarty T. (2000) Defining business rules-what are they really. In: Final Report

ISO (1987) Information processing systems–concepts and terminology for the conceptual schema and the information base

Kop C., Mayr H. C. (1998) Conceptual predesign bridging the gap between requirements and conceptual design. In: Proceedings of the 3rd International Conference on Requirements Engineering (ICRE'98). IEEE, pp. 90–98

Kop C., Mayr H. C., Yevdoshenko N. (2007) Requirements Modeling and MDA–Proposal for a Combined Approach. In: Advances in Information Systems Development, pp. 191–201

Lenzerini M., Santucci G. (1983) Cardinality Constraints in the Entity-Relationship Model. In: ER, pp. 529–549

Liddle S. W., Embley D. W., Woodfield S. N. (1993) Cardinality constraints in semantic data models. In: Data & Knowledge Engineering 11(3), pp. 235–270

Mayr H. C., Kop C. (2002) A User Centered Approach to Requirements Modeling. In: Proceedings of the Modellierung 2002. Lecture Notes in Informatics P-12 (LNI), GI-Edition, pp. 75–86

McAllister A. (1998) Complete rules for n-ary relationship cardinality constraints. In: Data & Knowledge Engineering 27(3), pp. 255–288

Morgan T. (2002) Business rules and information systems: aligning IT with business goals. Addison-Wesley Professional

Navathe S., Batini C., Ceri S. (1992) Conceptual Database Design–an Entity-Relationship Approach. In: Redwood City: Benjamin Cummings

OCL (2014) Object Constraint Language (OCL), Version 2.4

OMG (2001) Model Driven Architecture

OMG (2015) Unified Modeling Language (UML), version 2.5

OMG (2017) Semantics of Business Vocabulary and Business Rules, version 1.4

Paton N. W., Díaz O. (1999) Active database systems. In: ACM Computing Surveys (CSUR) 31(1), pp. 63–103

Ram S., Khatri V. (2005) A comprehensive framework for modeling set-based business rules during conceptual database design. In: Information Systems 30(2), pp. 89–118

Ramesh V., Ram S. (1997) Integrity constraint integration in heterogeneous databases: An enhanced methodology for schema integration. In: Information Systems 22(8), pp. 423–446

Ross R. (2005) Business Rule Concepts: Getting to the Point of Knowledge. Business Rule Solutions Inc.

Shao J., Pound C. (1999) Extracting business rules from information systems. In: BT Technology Journal 17(4), pp. 179–186

Simsion G. (2001) Data Modeling Essentials: Analysis, Design, and Innovation Scottsdale. In: AR, Coriolis

Storey V. C., Yang H.-L., Goldstein R. C. (1996) Semantic integrity constraints in knowledge-based database design systems. In: Data & Knowledge Engineering 20(1), pp. 1–37

Sweeney L. (2002) k-anonymity: A model for protecting privacy. In: International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10(05), pp. 557–570

Thalheim B. (1992) Fundamentals of cardinality constraints. In: International Conference on Conceptual Modeling. Springer, pp. 7–23

Thalheim B. (1996) An overview on semantical constraints for database models. In: Proceedings of the 6th International Conference Intellectual Systems and Computer Science

Umanath N. S., Scamell R. W. (2007) Data Modeling and Database Design. Thomson Course Technology

Vöhringer J., Mayr H. C. (2006) Integration of schemas on the pre-design level using the KCPM-approach. In: Advances in Information Systems Development. Springer, pp. 623–634

Von Halle B. (2001) Business rules applied: building better systems using the business rules approach. Wiley Publishing

Wand Y., Storey V. C., Weber R. (1999) An ontological analysis of the relationship construct in conceptual modeling. In: ACM Transactions on Database Systems (TODS) 24(4), pp. 494–528

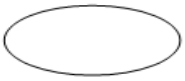
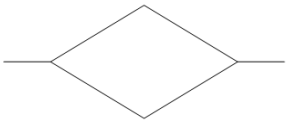
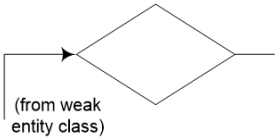
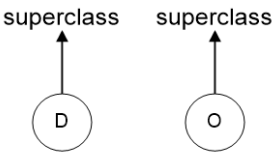
Wand Y., Weber R. (2002) Research commentary: information systems and conceptual modeling—a research agenda. In: Information systems research 13(4), pp. 363–376

Warmer J. B., Kleppe A. G. (2003) The object constraint language: getting your models ready for MDA. Addison-Wesley Professional

Warmer J., Kleppe A. (1999) The Object Constraint Language. Addison–Wesely

Weber R. (1996) Are attributes entities? A study of database designers' memory structures. In: Information Systems Research 7(2), pp. 137–162

## Appendix A : ER Model Syntax

| Symbol | Construct | Description |
|---|---|---|
|  | Entity Class | A set of entities for which common properties (attributes) are to be modeled |
|  | Regular Attribute | Properties shared by all members of an entity class. |
|  | Multi-valued Attribute | A single entity may have more than one value for such attributes |
|  | Identifying Attribute | An attribute that distinguishes one member entity from another |
|  | Partial Identifier | An attribute in a weak entity class used in conjunction with identifiers from strong entity classes for distinguishing member entities |
|  | Weak Entity Class | An entity class dependent on another (strong) entity class for its existence and part or all of its identifying attribute |
|  | Interaction Relationship | Association between members of one or more entity classes |
|  | Identifying Relationship | The arrow head denotes that a weak entity class depends on the relationship to determine its identifying classes |
|  | Inclusion Relationship<br>D : disjoint subclasses<br>O : overlapping subclasses | Relationship defining a generalization / specialization relationship between members of entity classes |

## Appendix B : Annotation Syntax in Backus-Naur Form

| | | |
|---|---|---|
| ⟨Construct Cardinality⟩ | ::= | ⟨Class Constraint⟩ \| ⟨Attribute / Domain Constraint⟩ \| ⟨Interaction Relationship Constraint⟩ |
| ⟨Class Constraint⟩ | ::= | CARD-C (⟨Entity Class⟩) |
| ⟨Attribute / Domain Constraint⟩ | ::= | CARD-A (⟨Entity Class⟩, Attribute) \| CARD-D (⟨Entity Class⟩, Attribute) |
| ⟨Attributes⟩ | ::= | Attribute, ⟨Attributes⟩ \| Attribute |
| ⟨Interaction Relationship Constraint⟩ | ::= | CARD-R-PT (⟨RE⟩) \| CARD-R-PT-SET (⟨RE⟩) \| CARD-R-PJ (⟨RE⟩) \| CARD-R-CO (⟨RE⟩, ⟨Entity Classes⟩) \| CARD-R-CO-SET (⟨RE⟩, ⟨Entity Classes⟩) |
| ⟨RE⟩ | ::= | Relationship, ⟨Entity Classes⟩ |
| ⟨Entity Classes⟩ | ::= | ⟨Entity Classes⟩, ⟨Entity Class⟩ \| ⟨Entity Class⟩ |
| ⟨Entity Class⟩ | ::= | Entity Class \| Role \| Role:⟨Class⟩ |
| ⟨Class⟩ | ::= | ⟨Entity Class⟩ \| Interaction Relationship Class |
| ⟨Card⟩ | ::= | ⟨min⟩:⟨max⟩ \| ⟨min⟩. . . ⟨max⟩ \| set-builder definition |
| ⟨min⟩ | ::= | Natural |
| ⟨max⟩ | ::= | Natural \| M |

# Big Data Conceptual Modelling in Cyber-Physical Systems

Ada Bagozi[a], Devis Bianchini[*,a], Valeria De Antonellis[a], Alessandro Marini[a], Davide Ragazzi[a]

[a] Dept. of Information Engineering, University of Brescia, Italy

Abstract. *Management of large volumes of data, collected from modern Cyber-Physical Systems, is calling for models, tools and methods for data representation and exploration, in order to capture relevant properties of physical objects, and manage them in the cyber-space. In this context, the impact of big data disruptive characteristics (namely, volume, velocity and variety) on data modelling and information systems design needs further investigation. In particular, data exploration is assuming an ever growing relevance, being a way users/operators can learn from data by inspecting it according to different perspectives. In this paper, we use conceptual modelling for (big) data exploration in a dynamic context of interconnected systems. We rely on a multi-dimensional model, that is suited for properly providing data organization for exploration. Furthermore, we propose a model-driven approach that guides the design of multiple exploration strategies according to different objectives. The model-driven approach exploits a model of relevance, aimed at focusing the attention of the users/operators only on relevant data that are being explored. We describe the instantiation of the proposed concepts through some scenarios in the smart factory context, in order to show how conceptual modelling helps abstracting from implementation details and focusing on semantics of explored data.*

Keywords. Data Exploration • Conceptual Modelling • Big Data • Multi-Dimensional Data • Industry 4.0 • Cyber-Physical Systems

## 1 Introduction

A Cyber-Physical System (CPS) is characterized by the integration of physical devices (machines and sensors) with cyber components (computer, data and programs) to form a context-sensitive system apt to react to dynamic changes in real-world (Lee and Seshia 2017). CPS are widespread in several domains like smart grids, autonomous automobile systems, domotics, medical monitoring and, more recently, Industry 4.0 (Lee et al. 2015b). Relevant function in CPS is the collection of raw data from dynamic physical environments, integrated with many types of cyber-space resources, and its transformation into actionable knowledge in real-time. Data management is leading to new CPS capabilities and challenges: for example, in the Industry 4.0 scenario, data is emerging as a new industrial asset, creating opportunities for operations improvement and increased industrial value through the capitalization of immaterial assets, and promoting advanced functions like self-awareness, self-configuration and self-repairing of machines (Hou and Wang 2013). To this aim, models, tools and methods for the collection, organization and exploration of data are required in order to capture relevant properties of physical objects, and manage them in the cyber space through information/data management and analysis systems (Monostori 2014). In particular, data exploration is assuming an ever growing relevance, being a way users/operators can learn from data by inspecting it according to

* Corresponding author.
E-mail. devis.bianchini@unibs.it

different perspectives. Nevertheless, the disruptive characteristics of big data, namely, volume, velocity and variety, pose additional issues for those who are in charge of extracting knowledge from it.

In this context, conceptual modelling can play a fundamental role, given its capability of abstracting data representation from its implementation in physical systems by means of concepts, their properties and mutual relationships (Chen 1976; Fliedl et al. 2005; Karagiannis et al. 2016; Olivé 2007), in order to build information systems (Cabot et al. 2017). Embley and Liddle (Embley and Liddle 2013) expect conceptual modelling to address big data challenges by structuring information, making big data volume searchable: it may help to highlight the semantics of underlying data in a fast and automatic way, choosing the best representation to foster data exploration. On the other hand, velocity of data acquisition requires the integration of different models and techniques, apt to properly summarize data that are incrementally collected from monitored interconnected systems.

In this paper, we propose a conceptual model apt to provide a high level representation of a Cyber-Physical System through a set of "facets" or "dimensions", either flat or hierarchically organized. Aggregation of data according to different dimensions (e.g., time, monitored system), being related to the observed physical problems, can give proper semantics to the collected data. Moreover, multi-dimensional model enables data exploration by following the hierarchical structure of dimensions. The proposed multi-dimensional model is integrated with data summarisation techniques, in order to provide a synthetic representation over large volumes of data to be managed. Given the conceptual model, we define a model-driven data exploration approach, that relies on data relevance techniques, aimed to focus the attention of the user/operator on relevant data only and to guide multiple exploration strategies according to different objectives. We envisage the application of the proposed model-driven approach to some paradigmatic research scenarios in the smart factory context (Lee et al. 2015a), in order to show how conceptual modelling helps abstracting from implementation details and focusing on semantics of explored data. In particular, the first scenario concerns monitoring of a Cyber-Physical System for anomaly detection and adaptive recovery from damage conditions. The second scenario shows the potential utility of the approach for data-driven performance comparison across different physical systems. The two scenarios are conceived for smart factory maintenance operators; in the first scenario, operators may be interested in preventing downtimes of the monitored systems, while in the second one operators may want to understand which part of the monitored system isn't working properly in order to replace or repair it. With respect to exploratory data analysis (Tukey 1977) and Data Mining (Han and Kamber 2006), our approach aims at supporting exploration as a multi-step process, where the users/operators may iteratively improve focus on relevant data, by receiving suggestions based on the model of relevance. Compared to On Line Analytical Processing (Golfarelli and Rizzi 2009), we manage data that is incrementally collected, organized and analysed on-the-fly. Finally, with respect to traditional faceted search (Tunkelang 2009), we deal with high data volumes and velocity, that imply efficient techniques for storing and managing them.

In (Bagozi et al. 2017c) we introduced the summarisation and relevance techniques as ingredients to perform exploration of real time data in a dynamic context of interconnected systems. In (Bagozi et al. 2017b) we proposed IDEAaS (Interactive Data Exploration As-a-Service), a framework where innovative services are designed to enable data exploration. In (Bagozi et al. 2017a) we discussed the application of the multi-dimensional model, data summarisation and relevance evaluation techniques to support anomaly detection in collaborative systems in the context of Cyber-Physical Systems and Industry 4.0. This paper extends this research for what concerns the introduction of a model-driven approach based on the conceptual model. In particular, we discuss the application of the approach for performance

comparison across different physical systems, introducing additional contributions with respect to anomaly detection issues and thus abstracting the characteristics of big data exploration over multiple scenarios.

The paper is organized as follows: Section 2 introduces the conceptual model with the help of a running example; Section 3 describes summarisation and relevance evaluation techniques engaged to support data exploration; in Section 4 we discuss the model-driven data exploration approach, taking into account different scenarios in the Industry 4.0 context; in Section 5 we highlight cutting-edge features of our approach compared to state of the art; finally, Section 6 closes the paper with some final remarks and future work.

## 2 Conceptual Model

### 2.1 Running example

To better explain concepts addressed in this paper, we will use the running example introduced in (Bagozi et al. 2017c). In the example, we considered an Original Equipment Manufacturer (OEM) producing multi-spindle machines for various industrial sectors: automotive, aviation, water industry, etc. A multi-spindle machine is a turning machine that allows multiple tools to cut pieces of material simultaneously and independently each other. A picture of the multi-spindle machine is shown in Figure 1. Each spindle is mounted on a unit moved by an electrical engine to perform X, Y, Z movements. The multiple spindles are carried in a precision rotating drum where raw material is positioned. The total number of operations needed to complete a manufacturing cycle are divided among the number of spindles, so that a cycle is completed with one full rotation of the drum. Each spindle is equipped by a cross-slide and end-slide tool. Tools are selected according to the instructions specified within the machine Part Program, that is, a set of instructions executed by the numerical control of the multi-spindle machine to manage its operations.

Spindle precision, working performances, minimization of tool breaks and machine downtimes are critical factors. Real time data collected from the multi-spindle machine concerns the spindle rotation as impressed by an electrical engine and its rotation speed as collected by the machine numerical control. For each spindle, we measure the velocity of the three axes (X, Y and Z) and the electrical current absorbed by each engine, the value of rpm (rotations per minute) for the spindle, the percentage of power absorbed by the spindle engine (charge coefficient). Hereafter, we will refer to the measured aspects as *features*. The aim of the OEM is to understand if it is possible to use real time data collected directly from the machine for monitoring the spindle rolling friction torque increase and the tool wear. With spindle rolling friction torque increase we refer to a specific behaviour of the spindle shaft that turns hard more and more due to different possible reasons: lack of lubrication and bearing wear that may lead to possible bearing failures. Tool wear monitoring is referred to possible tool usage optimizations in order to balance the trade-off between the number of tools used and the risk of breaking the tool during operations, that may lead to long downtimes. Spindle rolling friction torque increase and tool wear can be monitored by observing the spindle power absorption for similar rpm values. If a greater power absorption is detected, disregarding the tool that is being used, the spindle rolling friction torque increase could be identified as the possible anomaly that increases the energy request to perform the manufacturing operations. If the increase in absorbed power is related only to the usage of a particular tool, this can be recognized as a symptom of a possible exceeding tool wear. Therefore, machine components and tools, as well as time, represent multiple dimensions to perform data exploration. We will address such dimensions as first-class citizens of our conceptual model.

### 2.2 The model in a nutshell

The multi-dimensional model we propose for data exploration can be represented as an hypercube, as shown in Figure 2. In figure each node contains records of measures collected at a given time $t_1$. Measures are described through a timestamp and

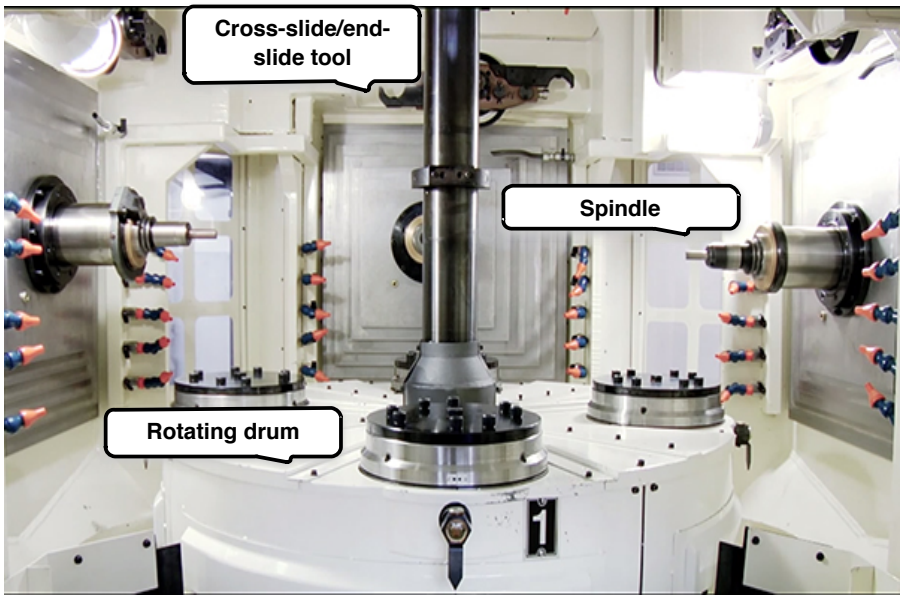



*Figure 1: Multi-spindle machine considered for the running example.*

the measured value. These measures are collected and organized according to several dimensions, such features/feature spaces and domain-specific-dimensions, representing hypercube axes.

**Features and feature spaces.** Measures are associated to `Features`, that is, the measured quantities, in turn collected into `Feature_Spaces`. A feature space conceptually represents a set of related features, that are jointly measured to observe a physical phenomenon. Multiple feature spaces might be observed, and the observation of a feature might be useful to monitor more than one feature space. In the considered running example, features are the speed over the three axes X, Y and Z, the electrical current, the value of spindle rpm and the percentage of absorbed power. The set composed of spindle power absorption and rpm features is an example of feature space used to monitor spindle rolling friction torque increase and tool wear. They can be observed by monitoring the spindle power absorption.

**Domain-specific dimensions.** These dimensions group together collected measures according to "facets", such as the observed machine or the tool used during manufacturing. Also domain-specific dimensions can be organized through hierarchies: tools can be aggregated into tool types (`Tool:Tool_type` axis in Figure 2), while monitored physical components (e.g., spindles) can be aggregated into the machines they belong to, in turn organized into plants and enterprises. Other dimensions might be the part program that is being executed by the numerical control of the monitored system and the working mode (G0, fast movement of the spindle, e.g., to catch the tool, or G1, slow movement of the spindle during the manufacturing). Among analysis dimensions, we always consider time.

In Figure 2 a subset of all possible dimensions is showed, achieved by slicing over the feature space $\texttt{fs}_1$ and the part program $\texttt{pp}_2$. For example, the node identified as `"A"` contains the records of measures collected at time $t_1$ for multi-spindle machine $\texttt{m}_1$ (spindle $\texttt{s}_3$), that is using tool $\texttt{u}_3$, during working mode $G_0$, considering features in the feature space $\texttt{fs}_1$, while running the part program $\texttt{pp}_2$.

Collected measures could be meaningfully compared over domain-specific dimensions: for example, it makes sense to compare measures across different components/machines, for performance

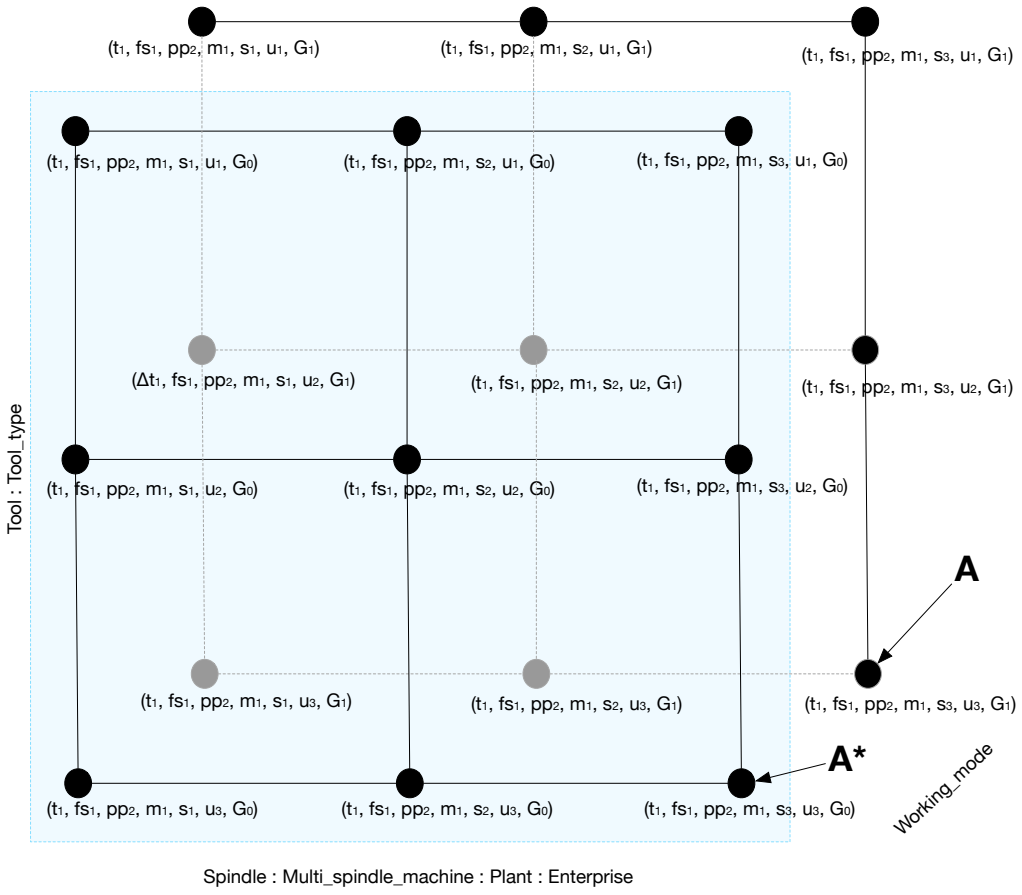



*Figure 2: The multi-dimensional data model for big data exploration.*

comparison purposes, or across different tools, in order to detect anomalies.

Users/operators can move over hypercube nodes for data exploration. In this work, we also introduce exploration constrains to prevent meaningless comparisons.

**Exploration constraints.** There are some characteristics that should remain constant while performing any kind of comparison between measures. For instance, comparing measures collected during working mode $G_0$ with those collected during working mode $G_1$ makes no sense. The same considerations hold for the tool type, indeed comparing measures collected from a spindle while using tools of different types doesn't lead to any conclusion. These considerations lead to the introduction of the concept of *Exploration constraints*, i.e., a set of dimensions, that may be at different hierarchical levels, over which comparison between measures does not make sense. An example of exploration constraint is the dimension `Tool` at the hierarchical level `Tool_type` or the dimension `Working_mode`. In Figure 2 node "A" represents measures that should not be compared to measures represented by node "A*", because the two nodes are related to measures collected in a different working mode $G_0$/$G_1$.

## 2.3 Model definitions

Formal definitions of measures and exploration dimensions are given in the following.

**Definition 1 (Feature)** A feature represents a monitored variable that can be measured. A feature $F_i$ is described as $\langle n_{F_i}, u_{F_i} \rangle$, where $n_{F_i}$ is the feature name, $u_{F_i}$ represents the unit of measure.

Let's denote with $F = \{F_1, F_2 \dots F_n\}$ the overall set of features.

**Definition 2 (Measure)** We define a measure for the feature $F_i$ as a scalar value $X_i(t)$, expressed in terms of the unit of measure $u_{F_i}$, taken at the timestamp $t$.

**Definition 3 (Feature space)** We denote with $FS = \{FS_1, FS_2, \dots FS_m\}$ the set of feature spaces, where $FS_j \subseteq F$. Given a feature space $FS_j = \{F_1, \dots F_h\}$, we denote with the vector $\vec{X}_j(t)$ a record of measures $\langle X_1(t), \dots X_h(t)\rangle$ for the features in $FS_j$, synchronized with respect to the timestamp $t$. Feature spaces can be monitored independently each others.

**Definition 4 (Domain-specific dimensions)** We denote with $\mathcal{D}$ the subset of the multi-dimensional space created by $p$ domain-specific dimensions $\mathcal{D}_1, \dots \mathcal{D}_p$, where $\mathcal{D} = \mathcal{D}_1 \times \dots \times \mathcal{D}_p$. Dimensions can be organized in hierarchies, at different levels. We denote with $\mathcal{D}_j^i$ the i-th level in the hierarchy of j-th dimension and with $d_i \in \mathcal{D}_i$ a single instance of the dimension $\mathcal{D}_i$.

**Definition 5 (Exploration constraints)** We define an exploration constraint $ECX_i$ as a tuple $(\mathcal{D}_j, i)$, where $i$ is the i-th level in the hierarchy over the j-th dimension $\mathcal{D}_j$. Comparison between different measures does not make sense over dimension $\mathcal{D}_j$ at the i-th hierarchical level. We denote with $ECX$ the set of all possible exploration constraints $\{ECX_i\}$.

**Definition 6 (Multi-dimensional model)** We describe the multi-dimensional model as a set $\mathcal{V}$ of nodes and a set of exploration constraints $ECX$. Each node $v \in \mathcal{V}$ is described as

$$v = \langle \vec{X}_j(t), fs_j, d_1, d_2, \dots d_p \rangle \qquad (1)$$

where $\vec{X}_j(t)$ represents a record of measures taken at time $t$, for the feature space $fs_j$ and the values $d_1, d_2, \dots d_p$ of domain-specific dimensions $\mathcal{D}_1, \dots \mathcal{D}_p$, $ECX$ represents the set of exploration constraints defined over the dimensions $\mathcal{D}_1, \dots \mathcal{D}_p$.

The conceptual model we defined to capture information collected from the Cyber-Physical System is shown in Figure 3 using ER notation. It reflects the typical structure of multi-dimensional models, where *facts* are represented by *measures*, as collected from the monitored physical system. Each feature presents physical limits (*bounds*), that should not be violated in order to avoid machine damages. We distinguish among warning and error bounds: (b) `warnings` identify anomalous conditions that may lead to breakdown or damage of machines; (c) `errors` identify unacceptable conditions in which a machine can not operate. Besides defining *features* bounds, we introduced the notion of *context* to specify contextual boundaries. A contextual bound represents the limit of a feature within a specific context where the feature is measured. The rationale is that, in a specific context (e.g., the working mode, the part program), when the Cyber-Physical System works normally, a feature should assume values within a specific range, that might be different from the overall physical limits for the same feature. These bounds are modelled through a relationship with attributes between the *feature* and the *context* entities in the model as shown in Figure 3.

Exploration constraints are not modelled in the conceptual model. They are defined as further meta-data that are used to guide/constrain the exploration.

## 3 Data summarisation and relevance evaluation

The characteristics of big data, namely, volume, velocity and variety, pose additional issues for data collection and organization. High volume calls for techniques and tools to provide a compact view over the large amount of collected data and to focus data exploration on relevant data only. Furthermore, when dealing with real time data, collected in Cyber-Physical Systems, data streams must be considered, where not all data are available since the beginning, but are collected in a fast and incremental way. To this aim, the conceptual

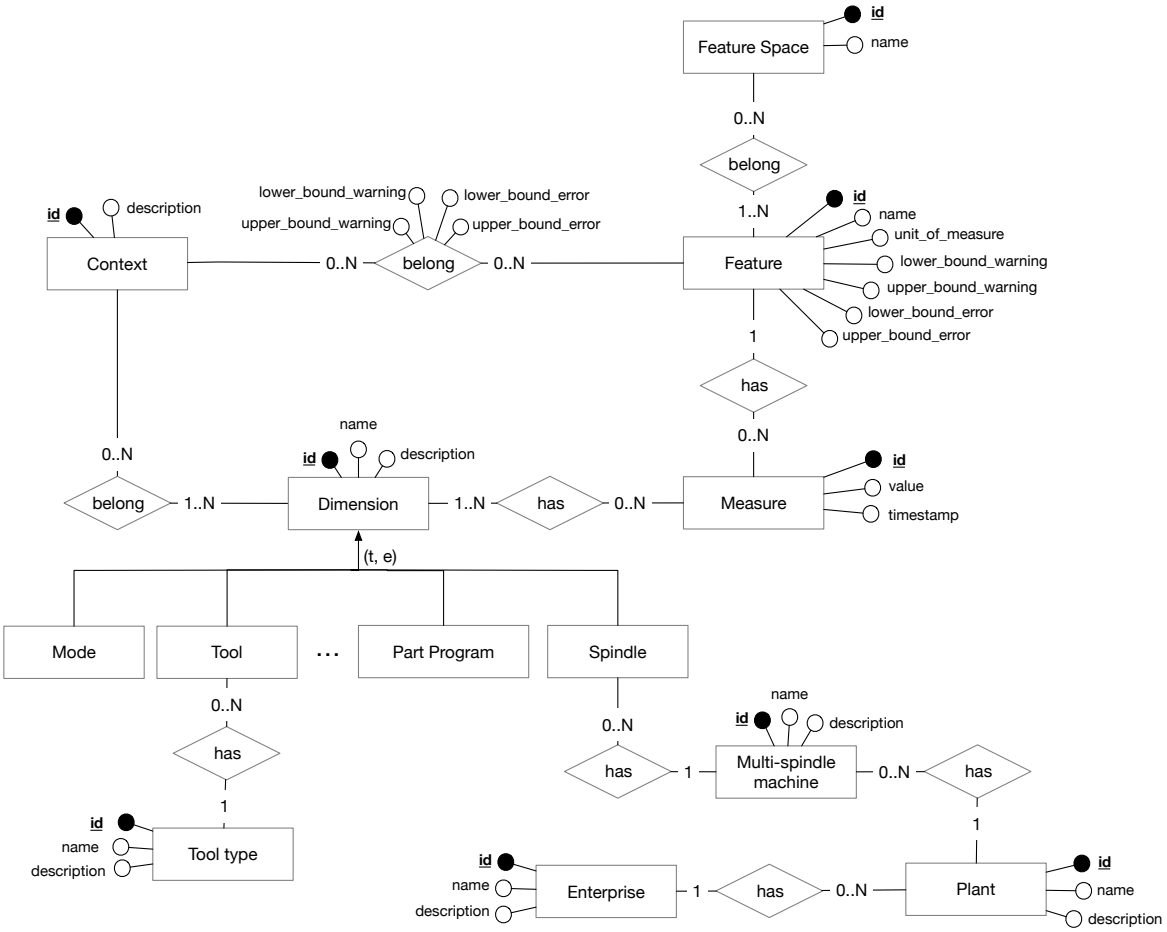



*Figure 3: Conceptual model for exploration of data collected from Cyber-Physical Systems.*

model proposed above is integrated with data summarisation and relevance evaluation techniques. These techniques have been detailed in (Bagozi et al. 2017c) and are shortly summarized in the following. The aim in this paper is to discuss the application of the conceptual model and relevance evaluation to different data exploration scenarios (see Section 4).

### 3.1 Clustering-based data summarisation

In our approach, data summarisation is based on clustering-based techniques. Clustering offers a two-fold advantage: (a) it gives an overall view over a set of measure records, using a reduced amount of information; (b) it allows to depict the behaviour of the system better than single records, that might be affected by noise and false outliers, in order to observe a given physical phenomenon.

The clustering algorithm is performed in two steps: (i) in the first one, a variant of Clustream algorithm (Aggarwal et al. 2003) is applied, that incrementally processes incoming data to obtain a *set of syntheses*; (ii) in the second step, X-means algorithm is applied (Pelleg and Moore 2000) in order to cluster syntheses obtained in the previous step. X-means algorithm does not require any a-priori knowledge on the number of output clusters. Syntheses are defined as follows.

**Definition 7 (Synthesis)** We define a synthesis $S_j$ in a feature space $fs_j$ as a tuple consisting of five elements, that is, $S_j = \langle N_j, \vec{LS}_j, SS_j, \vec{X}_j^0, R_j \rangle$, where: (i) $N_j$ is the number of records included

into the synthesis (from $\vec{X}_j(t_1)$ to $\vec{X}_j(t_N)$, where $t_N = t_1 + \Delta t$); (ii) $\vec{LS}_j$ is a vector representing the linear sum of measures in $S_j$; (iii) $SS_j$ is a scalar representing the quadratic sum of points in $S_j$; (iv) $\vec{X}_j^0$ is a vector representing the centroid of the synthesis; (v) $R_j$ is the radius of the synthesis. In particular:

$$\vec{LS}_j = \sum_{k=1}^{N_j} \vec{X}_j(t_k) \quad SS_j = \sum_{k=1}^{N_j} \vec{X}_j^2(t_k) \tag{2}$$

$$\vec{X}_j^0 = \frac{\sum_{k=1}^{N_j} \vec{X}_j(t_k)}{N_j} \tag{3}$$

$$R_j = \sqrt{\frac{\sum_{k=1}^{N_j} (\vec{X}_j(t_k) - \vec{X}_j^0)^2}{N_j}} \tag{4}$$

The second step aims to cluster syntheses. Clustering is performed to minimize the distance between syntheses centroids within the same cluster and to maximize the distance between syntheses centroids across different clusters. Clusters give a balanced view of the observed physical phenomenon, grouping together syntheses corresponding to the same working status.

**Definition 8 (Cluster)** A cluster $C$ is defined as follows: $C = \langle \vec{C}_0, \mathcal{S}_C \rangle$, where $\vec{C}_0$ is the cluster centroid, $\mathcal{S}_C$ is the set of syntheses belonging to the cluster. We denote with $SC$ the set of identified clusters.

An incremental data-stream clustering algorithm has been developed, where the clustering algorithm is computed incrementally over time. The minimum granularity of the time dimension corresponds to the time interval over which clustering is performed. This means that, considering $\Delta$t as the time interval on which records of measures are grouped in syntheses, that in turn are clustered, every $\Delta$t seconds the clustering algorithm outputs a new cluster set $SC$ built on top of the previous sets. $\Delta$t is chosen at configuration time such that $1/\Delta t$ is greater than the data acquisition frequency.

Let's denote with $\Sigma(\vec{X}_j(\Delta t), fs_j, d_1, d_2, \ldots d_p)$ the set of clusters obtained by applying clustering on measures collected during time interval $\Delta t$ for dimensions $d_1 \in \mathcal{D}_1$, $d_2 \in \mathcal{D}_2 \ldots$ $d_p \in \mathcal{D}_p$, for monitoring feature space $fs_j$; $\vec{X}_j(\Delta t)$ denotes the set of measures taken at a given time interval $\Delta t$, from $\vec{X}_j(t)$ to $\vec{X}_j(t + \Delta t)$. We include the output of summarisation procedure into the multi-dimensional model as follows, starting from Definition 6.

**Definition 9 (Cluster-based multi-dim. model)** We define the cluster-based multi-dimensional model as a set $\mathcal{V}'$ of nodes and a set of exploration constraints $ECX$. Each node $v' \in \mathcal{V}'$ is described as

$$v' = \langle \Sigma(\vec{X}_j(\Delta t), fs_j, d_1, d_2, \ldots d_p) \rangle \tag{5}$$

where $\Sigma(\cdot)$ represents the application of data summarisation procedure to $\vec{X}_j(\Delta t)$ for dimensions $d_1 \in \mathcal{D}_1$, $d_2 \in \mathcal{D}_2 \ldots$ $d_p \in \mathcal{D}_p$, for monitoring feature space $fs_j$; $ECX$ represents the set of exploration constraints defined over the dimensions $\mathcal{D}_1, \ldots \mathcal{D}_p$.

### 3.2 Data relevance evaluation

Relevance-based techniques are used to detect components status over time. In literature, data relevance is defined as the distance from an expected status. The point is to define the expected status and how to compute such a distance. In (Bagozi et al. 2017c) we defined the expected status as the set of clusters computed during normal working conditions for the monitored system, denoted with $\hat{SC}$, and data relevance is based on the notion of *cluster distance* between the clusters set $SC$, that represents the current behaviour of the monitored system, and the set $\hat{SC}$. In the following, we will describe how the conceptual model enables data relevance evaluation in different considered scenarios.

## 4 Model-driven data exploration approach

Data exploration is performed on top of the multi-dimensional model in order to pursue different

goals. In particular, in this section we discuss two possible exploration scenarios:

- *exploration for anomaly detection*, to promptly identify anomalies by monitoring and observing if collected data overtakes or gets closer to feature or contextual bounds, that represent physical limits of breakage of the monitored system; this kind of exploration may be implemented in a state detection service, and used by OEMs to prevent downtimes of monitored systems and by multi-spindle machine owner to plan supply chain activities;
- *exploration for performance comparison*, to compare performances of different monitored systems, while fixing the other analysis dimensions (e.g., using the same tools, performing the same manufacturing steps as codified within the part program); this kind of exploration can be used by OEMs to monitor a machine fleet over multiple clients and to offer remote configuration and optimization services.

Beyond these options, generic exploration of collected data is enabled by the multi-dimensional model. We assume that the user/operator formulates an explicit, albeit vague exploration request, by instantiating a subset of available dimensions (e.g., a specific spindle, tool or part program), and expects the system to suggest some promising data to explore. Data summarisation techniques contribute to reduce the complexity of the exploration by providing a compact view over underlying data. In the following, we will focus on the two exploration scenarios mentioned above, as generic exploration has been already described in (Bagozi et al. 2017c), where traversals inspired by operators in OLAP systems have been proposed for browsing data within the multi-dimensional space. We remark that the list of scenarios we will discuss here is not exhaustive. Our aim is to show how the conceptual model's features can be properly used to support model-driven data exploration mechanisms and can be configured for different scenarios.

### 4.1 Exploration for anomaly detection

The goal of a state detection service is to detect anomalies and send alerts concerning the system status. We consider three different values for the *status*: (a) `ok`, when the system works normally; (b) `warning`, when the system works in anomalous conditions that may lead to breakdown or damage; (c) `error`, when the system works in unacceptable conditions or does not operate. The warning status is used to perform an early detection of a potential deviation towards an error state. The migration of the system status from one value to the others raises an *alert* and occurs when one or more measures exceed a given bound, either a feature bound or a contextual bound, according to the conceptual model definitions given in Section 2. These bounds set the ranges for the three different values of the status: `ok`, `warning` and `error`. Feature bounds determine the *absolute status* of a feature, while contextual bounds determine the *contextual status* of a feature. The system status (either absolute or contextual) can be propagated to the whole feature space and along the hierarchy of monitored physical system, according to the following propagation rules.

a) *Propagation over the feature space.* Given a feature space $FS_j = \{F_1, F_2, \ldots F_h\}$, the value of the status associated to $FS_j$, given the status values for each feature $F_1, F_2, \ldots F_h$, is computed as follows:
   - `ok`, if the status of each feature $F_i(\forall i = 1 \ldots h)$ is `ok`;
   - `warning`, if the status of at least one feature $F_i(\forall i = 1 \ldots h)$ is `warning`;
   - `first level error`, if the status of at least one feature $F_i(\forall i = 1 \ldots h)$ is `error`;
   - `second level error`, if the status of each feature $F_i(\forall i = 1 \ldots h)$ is `error`.

b) *Propagation along the hierarchy of the monitored physical system.* The value of a feature status for a spindle is propagated to the highest

level of the hierarchy (machine, plant, enterprise) as follows: the status of the machine is

- `ok`, if the status for all its spindles is `ok`;
- `warning`, if the status of at least one spindle is `warning`;
- `first level error`, if the status of at least one spindle is `error`;
- `second level error`, if the status of all its spindles is `error`.

The same applies for the status of the plant (resp., enterprise), computed starting from the status of its machines (resp., plants).

**Relevance-based anomaly detection.** For anomaly detection, the expected status is identified through the set $\hat{SC} = \{\hat{C}_1, \hat{C}_2, \ldots \hat{C}_n\}$ of clusters computed during normal working conditions. Relevant data are recognized when their clusters set differs from $\hat{SC}$. Let's denote with $SC = \{C_1, C_2, \ldots, C_m\}$ the current clusters set, where $n$ and $m$ do not necessarily coincide. We evaluate the distance between $SC$ and $\hat{SC}$ by aggregating distances between each cluster belonging to $SC$ and the closest cluster belonging to $\hat{SC}$ and vice-versa, for symmetry purposes. Formally, the distance is computed as:

$$\Delta(SC, \hat{SC}) = \frac{\sum_{i=1}^{m} d(C_i, \hat{SC}) + \sum_{j=1}^{n} d(SC, \hat{C}_j)}{m+n} \tag{6}$$

where $d(C_i, \hat{SC}) = min_{j=1,\ldots n} d_c(C_i, \hat{C}_j)$ and $d(SC, \hat{C}_j) = min_{i=1,\ldots m} d_c(C_i, \hat{C}_j)$ is the distance between clusters. To compute the distance between two clusters $d_c(C_i, \hat{C}_j)$, we combined different factors: (i) the distance between clusters centroids $d_{\vec{C}_0}(C_i, \hat{C}_j)$, to verify if $C_i$ translates with respect to $\hat{C}_j$; (ii) the *intra-cluster distance* $d_c^{intra}(C_i, \hat{C}_j)$, to verify if there has been an expansion or a contraction of cluster $C_i$ with respect to $\hat{C}_j$; (iii) the difference in number of syntheses contained in $C_i$ and $\hat{C}_j$, denoted with $d_N(C_i, \hat{C}_j)$. The overall value of $d_c(C_i, \hat{C}_j)$ is given by:

$$\begin{aligned} d_c(C_i, \hat{C}_j) = \alpha \cdot d_{\vec{C}_0}(C_i, \hat{C}_j) \\ +\beta \cdot d_c^{intra}(C_i, \hat{C}_j) \\ +\gamma \cdot d_N(C_i, \hat{C}_j) \end{aligned} \tag{7}$$

where $\alpha$, $\beta$ and $\gamma \in [0, 1]$ are weights such that $\alpha + \beta + \gamma = 1$, used to balance the impact of terms in Equation (7). To set the optimal weights, a grid procedure can be performed over $\alpha$ and $\beta$ ($\gamma$ is set with $1 - \alpha - \beta$), with the value of each weight varying from 0 to 1. In our preliminary experiments, presented in (Bagozi et al. 2017c), we put $\alpha = \beta = \gamma = \frac{1}{3}$.

In particular, $d_{\vec{C}_0}(C_i, \hat{C}_j)$ is computed by applying the Euclidean distance ($D0$) between clusters centroids, according to the following formula:

$$D0 = \sqrt{(\vec{C}_0^i - \hat{\vec{C}}_0^j)^2} \tag{8}$$

where $\vec{C}_0^i$ and $\hat{\vec{C}}_0^j$ are centroids of $C_i$ and $\hat{C}_j$, respectively. The computation of intra-cluster distance $d_c^{intra}(C_i, \hat{C}_j)$, performed on the sets of syntheses of $C_i$ and $\hat{C}_j$, is similar to the computation of $\Delta(SC, \hat{SC})$ in Equation (6), that is:

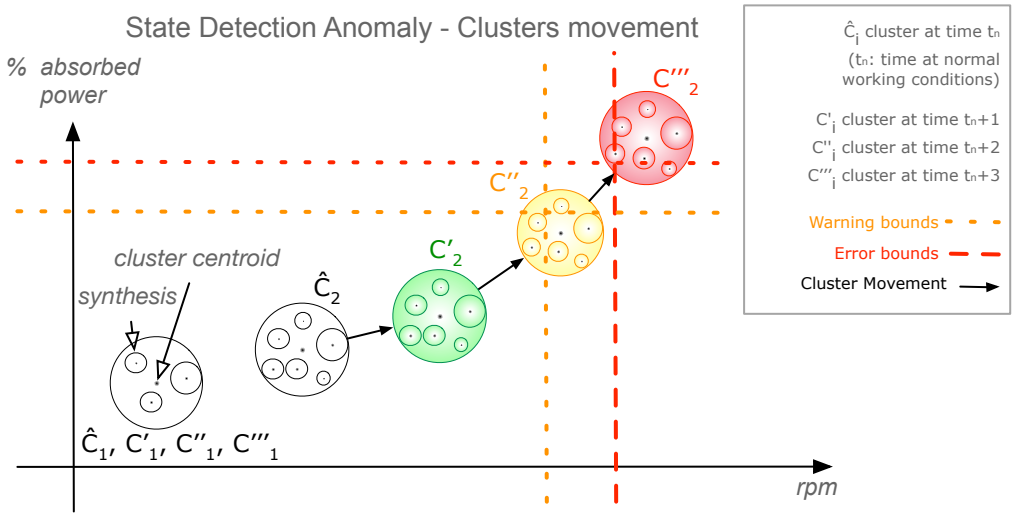
$$d_c^{intra}(C_i, \hat{C}_j) = \frac{\sum_{k=1}^{n_1} d_s(S_k, \hat{C}_j) + \sum_{h=1}^{n_2} d_s(C_i, S_h)}{n_1 + n_2} \tag{9}$$

where $S_k \in \mathcal{S}_{C_i}$, $S_h \in \mathcal{S}_{\hat{C}_j}$, $|\mathcal{S}_{C_i}| = n_1$, $|\mathcal{S}_{\hat{C}_j}| = n_2$, $d_s(S_k, \hat{C}_j) = min_{h=1,\ldots n_2} d_s(S_k, S_h)$ and $d_s(C_i, S_h) = min_{k=1,\ldots n_1} d_s(S_k, S_h)$. Term $d_s(S_k, S_h)$ represents the average inter-syntheses distance ($D1$):

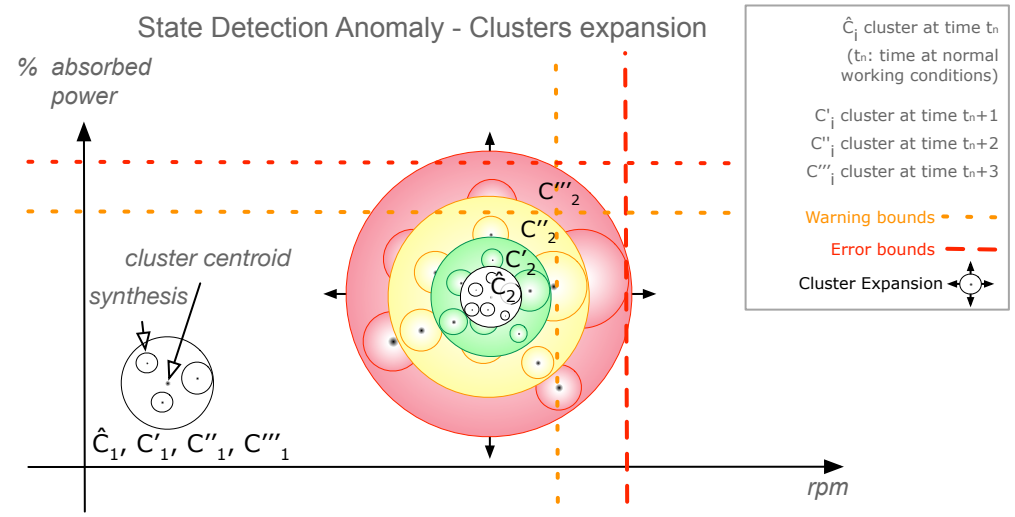
$$D1 = \sqrt{\frac{\sum_{i=1}^{N1} \sum_{j=N1+1}^{N1+N2} (\vec{X}(t_i) - \vec{X}(t_j))^2}{N1N2}} \tag{10}$$

where $N1$ and $N2$ are the number of records in $S_k$ and $S_h$, respectively.

The proposed relevance techniques enable to detect over time clusters movements, clusters contraction/expansion, changes in the number of clusters. Figures 4(a) and (b) show examples




(a)




(b)

*Figure 4: Illustration of clusters sets changing over time for anomaly detection: (a) clusters movements; (b) clusters expansion. Clusters set $\hat{SC}$ is generated at time $t_n$ at normal working conditions and consists of clusters $\hat{C}_1$ and $\hat{C}_2$.*

of clusters changes for the anomaly detection purpose: changes in clusters set over time is detected due to spindle rolling friction torque increase, causing a decrease of rpm and an increase of the percentage of absorbed power. This exploration scenario is applied by fixing all the dimensions and observing evolution over time of measures within the feature space. In particular, the relevance techniques allow to identify what are the clusters that changed over time. Let's denote with $\{\overline{C_i}\}$ the set of such clusters. Data that is summarized in the clusters $\{\overline{C_i}\}$ is considered as relevant and, for each cluster in $\{\overline{C_i}\}$, the distance of cluster centroid from the warning and error bounds is computed. If this distance is equal or lower than the cluster radius, this means that a warning or error status has to be detected. Note that distance also helps to detect *potential* state

changes. Consider for example Figure 4(a), that shows an example of clusters evolution over time for the smart factory case study. The figure shows how the cluster $C_1$ doesn't changed its position, as well as its size, from time $t_n$ to $t_{n+3}$. On the other hand, cluster $C_2$ evolves from the wealth zone to the warning and error zones. At time $t_{n+2}$ cluster $C_2''$ crosses the warning bound of rpm feature causing a warning alert, at time $t_{n+3}$ cluster $C_2'''$ moves into the error zone, crossing error bounds of both the considered features. At time $t_{n+1}$ cluster $C_2'$ still remains inside the wealth zone, however relevance techniques detected its change. Therefore cluster $C_2'$ is recognised as relevant and monitored to detect warning or error state changes. This allows for better performance of the anomaly detection algorithm, that focuses only on potential state changes.

### 4.2 Exploration for performance comparison

The goal of this kind of exploration is to compare different machines in order to identify changes in working conditions. Therefore, this exploration scenario is applied by comparing clusters sets over the monitored system dimension and fixing all the other domain-specific dimensions. Let's consider the situation depicted in Figure 5. In the figure, two machines are compared considering how the clusters sets distance between two spindles evolves over time. At time $t_1$ the two spindles present a clusters sets distance equal to $d_1$. This distance is usually different from 0 since the two spindles work in different environments and the likelihood of having exactly the same measures for considered features over the compared physical systems is very low. We refer to distance $d_1$ as *baseline distance*, occurring when all the spindles are working in normal conditions. The baseline distance can be computed as follows:

$$\Delta_{baseline}(\mathcal{M}_1, \mathcal{M}_2) = \Delta(\hat{SC}_1, \hat{SC}_2) \qquad (11)$$

where $\mathcal{M}_1$ and $\mathcal{M}_2$ are the two considered spindles, $\hat{SC}_1$ (resp., $\hat{SC}_2$) is the clusters set obtained for spindle $\mathcal{M}_1$ (resp., $\mathcal{M}_2$) during normal working conditions.

At time $t_{n+1}$ the two spindles present a distance $d_2 = d_1$. This means that the relative distance in terms of clusters sets between the two spindles is not changed. We remark here that, as shown in Figure 5, the condition $d_2 = d_1$ holds also if the two spindles changed their behaviours, and their respective clusters sets evolved accordingly. On the other hand, at time $t_{n+2}$ the distance $d_3$ is changed compared to $d_1$ and $d_2$, meaning that the two spindles started behaving differently each other. The metric of relevance, in this case, aims at highlighting the difference between $d_i$ and the baseline distance, denoted as $\Delta_{t_{n+2}}(\mathcal{M}_1, \mathcal{M}_2)$ and computed as follows:

$$\Delta(SC_1, SC_2) - \Delta_{baseline}(\mathcal{M}_1, \mathcal{M}_2) \qquad (12)$$

where $SC_1$ (resp., $SC_2$) is the clusters set obtained at time $t_{n+2}$ for spindle $\mathcal{M}_1$ (resp., $\mathcal{M}_2$). These considerations raise some additional questions, namely: (a) what about if we consider more than two spindles? (b) what about if we are observing two spindles whose behaviour evolves accordingly (case $d_2$) towards anomalous conditions? (c) what are the conditions under which we can also identify what are the spindles whose performances decreased compared to the other one? For what concerns the latter question, consider the case $d_3$: what is the spindle with decreased performances among the two observed ones?

To solve the first question, we provide an extension of Equations (11) and (12): $\Delta_{baseline}(\mathcal{M}_i, \mathcal{M}_j)$ and $\Delta_{t_k}(\mathcal{M}_i, \mathcal{M}_j)$ are computed for each pair $\mathcal{M}_i$ and $\mathcal{M}_j$ of compared spindles and the average values are considered.

The second question can be answered by combining together different exploration scenarios: in case of distance $d_2 = d_1$, exploration for anomaly detection applied on one of the considered spindles might help to identify the case in which all spindles changed their behaviour accordingly. Finally, for what concerns the third question, the

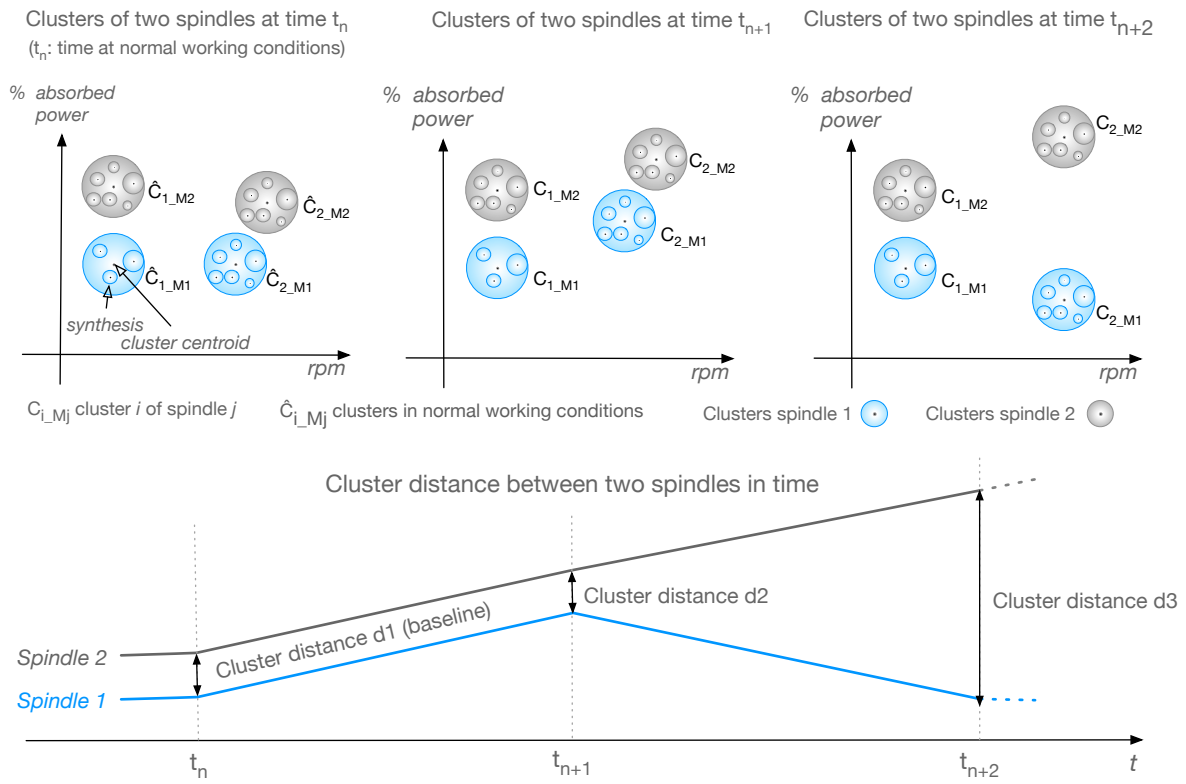



*Figure 5: Illustration of clusters sets changing over time for performance comparison. $\hat{C}_{i\_\mathcal{M}j}$ represents a cluster for spindle $\mathcal{M}_j$ while working in normal conditions ($j = 1, 2$).*

identification of the spindle with decreased performances with respect to the other ones can be detected by applying anomaly detection on each spindle or, if we are considering more than two spindles, according to the "two out of the three" logics: the target spindle will present a distance from the other ones that is greater than the same computation made for all the spindles. In case of two spindles, the third question can be answered only by applying exploration for anomaly detection for all the considered spindles.

### 4.3 Discussion

Our work proposes a conceptual modelling approach to allow flexible big data exploration in Cyber-Physical Systems, specifically to enable detection of unexpected situations. We investigated the potential benefits of conceptual modelling to foster exploration scenarios that pursue different goals. In particular, we considered two scenarios in the Industry 4.0 context: exploration for anomaly detection, to promptly identify anomalies on the monitored Cyber-Physical System, and exploration for performance comparison across different Cyber-Physical Systems. Our examples show how conceptual modelling reveals to be useful in abstracting from specific characteristics of each scenario:

- it enables to specify the desired exploration dimensions (e.g., features/feature space, or domain-specific dimensions as spindles, tools, etc.) thanks to the multi-dimensional paradigm; for example, exploration for anomaly detection is applied by observing over time the evolution of measures collected on monitored features and fixing all the other dimensions; while exploration for performance comparison is applied by comparing clusters sets across different mon-

itored systems and fixing all the other domain-specific dimensions;

- it is used to adapt the relevance techniques to the different cases, starting from an expected status and using the clusters sets distance analysis to identify changes: for example, in the case of exploration for anomaly detection, clusters sets distance analysis is based on the distance with respect to the clusters set $\hat{SC}$, generated at normal working conditions of the monitored system; in the exploration for performance comparison, clusters sets distance analysis is based on a baseline distance, obtained by considering normal working conditions of all the compared physical systems.

The abstraction enabled by conceptual modelling suggests the feasibility to provide a model-driven framework, where models and techniques can be adapted to different domains, beyond the Industry 4.0 one considered here.

## 5 Related Work

In this section we analyse some approaches that have been focused on data exploration and exploratory computing research fields, investigating the potential use of conceptual modelling to meet their goals. This comparison is summarized in Table 1.

The approach presented in (Kamat et al. 2014) deals with structured, multi-dimensional OLAP data, incrementally collected and organized in a cube structure, where axes correspond to facets to guide the exploration. The faceted cube exploration model is used to bound the space of possible queries. Data sampling techniques are applied to guess the next choices the user will likely to perform, thus reducing response times.

The approach presented in (Kalinin et al. 2014) treats multi-dimensional data, but it does not provide a conceptual modelling approach for exploration. The approach enables range queries (explicitly formulated by the user) on features, that must have a sortable numeric data type. Queries identify partially overlapping windows, shown to the user according to a cost-benefit criterion, that depends on the efforts required to collect data shown in the windows.

Also the approach in (Dimitriadou et al. 2016) handles multi-dimensional data already available for data exploration, but it does not provide a reference conceptual framework apt to be applied to different scenarios. The data shown to the user is fetched from the DBMS using sampling techniques. This approach builds clusters of objects using data mining techniques (k-means) and applies a classifier to infer data relevance.

The approach presented in (Costa et al. 2017) relies on multi-dimensional data incrementally collected. Moreover, a loss-less compression algorithm is used in order to minimise space consumption. In order to retrieve interesting events, authors exploit an occurrence frequency threshold: values whose frequency is below such threshold are properly highlighted.

Other approaches use multi-dimensional model to organize data, thus demonstrating the effectiveness of this kind of model to enable data exploration. Some of them also use summarisation/approximation techniques, to provide compact views over data and relevance evaluation techniques in order to guide exploration. The main contribution of our work compared to them mainly resides on the abstraction we performed in order to adapt the model and techniques to different scenarios. Existing approaches either do not mention any specific application scenario (making the proposal difficult to apply in a specific context such as the one of Cyber-Physical Systems) or are focused on very specific problems such as anomaly detection (Huber et al. 2016; Moghaddass and Zuo 2014; Stojanovic et al. 2016; Wang and Agrawal 2011). For what concerns anomaly detection approaches, the investigation of multi-dimensional modelling with summarisation and relevance evaluation techniques is limited. We refer to (Bagozi et al. 2017a) for a survey on this kind of approaches.

## 6 Concluding remarks

In this paper we discussed the application of conceptual modelling to provide a high level big data

| | IDEAaS | [Kamat et al. 2014] | [Kalinin et al. 2014] | [Dimitriadou et al. 2016] |
|---|---|---|---|---|
| Multi-dimensional model | ✓ | ✓ | ✓ | ✓ |
| Multi-dimensional model construction | ✓ | ✓ | | |
| Summarisation techniques | ✓ | | | ✓ |
| Approximation techniques | | ✓ | ✓ | ✓ |
| Relevance techniques | ✓ | | | ✓ |

| | IDEAaS | [Costa et al. 2017] | [Stojanovic et al. 2016] | [Wang and Agrawal 2011] |
|---|---|---|---|---|
| Multi-dimensional model | ✓ | ✓ | | ✓ |
| Multi-dimensional model construction | ✓ | (✓) | | |
| Summarisation techniques | ✓ | ✓ | ✓ | |
| Approximation techniques | | | | ✓ |
| Relevance techniques | ✓ | ✓ | | |

| | IDEAaS | [Huber et al. 2016] | [Moghaddass and Zuo 2014] | |
|---|---|---|---|---|
| Multi-dimensional model | ✓ | | ✓ | |
| Multi-dimensional model construction | ✓ | | | |
| Summarisation techniques | ✓ | ✓ | | |
| Approximation techniques | | | | |
| Relevance techniques | ✓ | | | |

*Table 1: Overview of approaches on (big) data exploration.*

representation and enabling model-driven data exploration. In particular, we exploited the ability of conceptual modelling to abstract data representation from implementation details and to focus on data semantics, by considering multiple exploration scenarios for monitoring Cyber-Physical Systems. We proposed a data representation model structured over a set of dimensions, hierarchically modelled. The resulting multi-dimensional model has been further enriched with data summarisation techniques, to provide a compact view over large amount of data and therefore managing data complexity in terms of volume and acquisition speed (velocity). Given the conceptual model, we defined a model-driven data exploration approach, that relies on data relevance techniques, aimed to focus the attention of the operators on relevant data only and to guide multiple exploration strategies according to different objectives. We also introduced exploration constraints to prevent useless comparison between collected data.

Further research will be performed in order to expand the set of analysis dimensions, in order to properly correlate data collected from CPS with high level aspects, concerning product and process quality, energy consumption and manufacturing

sustainability. The research can fruitfully exploit the flexibility of the proposed conceptual model. Additional scenarios will be also considered, and a model-driven tool to support configuration and set up of new scenarios will be investigated. Finally, future work will also be focused on analysing data visualization techniques, already addressed in approaches like (Kruiger et al. 2017) and (Saket et al. 2017), and developing a proper GUI specifically meant for big data exploration.

## References

Aggarwal C., Han J., Wang J., Yu P. (2003) A framework for clustering evolving data streams. In: Proc. of 29th International Conference on Very Large Data Bases (VLDB), pp. 81–92

Bagozi A., Bianchini D., De Antonellis V., Marini A., Ragazzi D. (2017a) Big Data Summarisation and Relevance Evaluation for Anomaly Detection in Cyber Physical Systems. In: On the Move to Meaningful Internet Systems. OTM 2017 Conferences, pp. 429–447

Bagozi A., Bianchini D., De Antonellis V., Marini A., Ragazzi D. (2017b) Interactive Data Exploration as a Service for the Smart Factory. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 293–300

Bagozi A., Bianchini D., De Antonellis V., Marini A., Ragazzi D. (2017c) Summarisation and Relevance Evaluation Techniques for Big Data Exploration: the Smart Factory case study. In: Proc. of 29th Int. Conference on Advanced Information Systems Engineering (CAISE2017), pp. 264–279

Cabot J., Gómez C., Pastor O., Sancho M., Teniente E. (2017) Conceptual Modeling Perspectives. Springer

Chen P. (1976) The entity-relationship model - toward a unified view of data. In: ACM Transactions on Database Systems 1(1), pp. 9–36

Costa C., Chatzimilioudis G., Zeinalipour-Yazti D., Mokbel M. F. (2017) Efficient Exploration of Telco Big Data with Compression and Decaying. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 1332–1343

Dimitriadou K., Papaemmanouil O., Diao Y. (2016) AIDE: An Active Learning-Based Approach for Interactive Data Exploration. In: IEEE Transactions on Knowledge and Data Engineering 28(11), pp. 2842–2856

Embley D., Liddle S. (2013) Big Data - Conceptual Modeling to the Rescue. In: Proc. of International Conference on Conceptual Modeling (ER), pp. 1–8

Fliedl G., Kop C., Mayr H. (2005) From textual scenarios to a conceptual schema. In: Data & Knowledge Engineering 55, pp. 20–37

Golfarelli M., Rizzi S. (2009) Data Warehouse Design: Modern Principles and Methodologies. McGraw-Hill

Han J., Kamber M. (2006) Data Mining: Concepts and Techniques Edition 2. (ed.). Morgan Kaufmann Publisher

Hou Z., Wang Z. (2013) From model-based control to data-driven control: Survey, classification and perspective. In: Information Science 235, pp. 3–25

Huber M., Voigt M., Ngonga Ngomo A.-C. (2016) Big Data Architecture for the Semantic Analysis of Complex Events in Manufacturing. In: INFORMATIK 2016 - the 46th Conference of the Gesellschaft für Informatik e.V. (GI), 2nd International Workshop on Big Data, Smart Data and Semantic Technologies (BDSDST), pp. 352–360

Kalinin A., Cetintemel U., Zdonik S. (2014) Interactive data exploration using semantic windows. In: Proc. of the ACM SIGMOD International Conference on Management of Data, pp. 505–516

Kamat N., Jayachandran P., Tunga K., Nandi A. (2014) Distributed and Interactive Cube Exploration. In: Proc. of 30th International Conference on Data Engineering (ICDE), pp. 472–483

Karagiannis D., Mayr H., Mylopoulos J. (2016) Domain-Specific Conceptual Modeling - Concepts, Methods and Tools. Springer International Publishing, pp. 1–594

Kruiger J. F., Hassoumi A., Schulz H.-J., Telea A., Hurter C. (2017) Multidimensional Data Exploration by Explicitly Controlled Animation. In: Informatics 4(3), pp. 1–21

Lee E. A., Seshia S. A. (2017) Introduction to Embedded Systems, A Cyber-Physical Systems Approach, Second Edition. MIT Press, ISBN 978-0-262-53381-2

Lee J., Ardakani H., Yang S., Bagheri B. (2015a) Industrial big data analytics and cyber-physical systems for future maintenance and service innovation. In: Proc. of Conference on Intelligent Computation in Manufacturing Engineering (CIRP) Vol. 38, pp. 3–7

Lee J., Bagheri B., Kao H. (2015b) A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. In: Manufacturing Letters 3, pp. 18–23

Moghaddass R., Zuo M. J. (2014) An integrated framework for online diagnostic and prognostic health monitoring using a multistate deterioration process. In: Reliability Engineering & System Safety 124(Supplement C), pp. 92–104

Monostori L. (2014) Cyber-physical production systems: Roots, expectations and R&D challenges. In: Proc. of the 47th CIRP Conference on Manufacturing Systems, pp. 9–13

Olivé A. (2007) Conceptual Modeling of Information Systems. Springer-Verlag Berlin Heidelberg

Pelleg D., Moore A. (2000) X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In: Proc. of 17th International Conference on Machine Learning (ICML), pp. 727–734

Saket B., Kim H., Brown E. T., Endert A. (2017) Visualization by Demonstration: An Interaction Paradigm for Visual Data Exploration. In: IEEE Transactions on Visualization and Computer Graphics 23(1), pp. 331–340

Stojanovic L., Dinic M., Stojanovic N., Stojadinovic A. (2016) Big-data-driven anomaly detection in industry (4.0): An approach and a case study. In: 2016 IEEE International Conference on Big Data (Big Data), pp. 1647–1652

Tukey J. (1977) Exploratory data analysis. Addison-Wesley Publishing Company Reading, pp. 1–688

Tunkelang D. (2009) Faceted Search (Synthesis Lectures on Information Concepts, Retrieval and Services). Morgan and Claypool Publishers, pp. 1–94

Wang F., Agrawal G. (2011) Effective and Efficient Sampling Methods for Deep Web Aggregation Queries. In: Proc. of Conference on Extending Database Technology (EDBT), pp. 425–436

# Simulation of Multi-Stage Industrial Business Processes Using Metamodelling Building Blocks with ADOxx

Dominik Bork[a], Hans-Georg Fill[b], Dimitris Karagiannis[a], Wilfrid Utz[*,a]

[a] Research Group Knowledge Engineering, Faculty of Computer Science, University of Vienna, Austria
[b] Information Systems, System Development and Database Application Group, Faculty of Information Systems and Applied Computer Sciences, University of Bamberg, Germany

Abstract. *This paper introduces Industrial Business Process Management (IBPM) as a novel research direction for information science (IS) based on the European Commission's project GO0D MAN. The project's aim is to establish a knowledge-based, ICT-supported approach for IBPM using system's engineering and optimization techniques realized by hybrid conceptual modelling methods. The contribution of this paper is a novel procedural framework that guides the design and development of such hybrid modelling methods. The framework comprises three abstraction levels:* a) abstract metamodel building blocks – *to define the abstract modelling language constructs and model processing capabilities required for domain aspects (e.g., in manufacturing: multi-stage, material, information and control flows);* b) model & functional building blocks - *as concrete modelling structures and analytical functionalities processing the models; and* c) execution building blocks - *a corresponding modelling and model processing environment implementation to support the modeller during the application. Composition and injection mechanisms on abstract building blocks enable efficient realization of concrete modelling and model processing capabilities by re-using and/or extending existing artefacts. Evaluation is performed by using the metamodelling platform ADOxx for a proof-of-concept implementation of a multi-stage manufacturing process simulation environment.*

Keywords. Business Process Management • Simulation • Analysis • Industrial Case • Metamodelling • Method Engineering

## 1 Introduction

Management of industrial manufacturing environment systems has seen a continuous adaptation and change in the past years as a result of the ongoing evolution in the context of the 4th industrial revolution coined Industry 4.0. Advanced manufacturing processes, flexible information and communication technologies, and involvement of knowledge workers require for modelling, simulation and forecasting techniques as enablers to contribute to the challenges of the manufacturing industry in the future (European Commission 2013). This paper presents an integrated, tool-supported, knowledge management system that enables the modelling, simulation and analysis of manufacturing systems by utilizing a conceptual modelling method, as defined in (Karagiannis et al. 2016b), enriched with a multi-stage simulation capability. The herewith created conceptual models convert tacit to explicit knowledge and enable knowledge processing (cf. (Cairó Battistutti and Bork 2017)).

* Corresponding author.
E-mail. wilfrid.utz@univie.ac.at
The Agent Oriented Zero Defect Multi-Stage Manufacturing (GO0D MAN) project under the coordination of Dr. Cristina Cristalli has received funding from the European Commission under the EU Framework Programme for Research and Innovation Horizon 2020 (2014 - 2020) within the FoF – Technologies for Factories of the Future initiative. Contract no. H2020-FOF-03-2016-723764. The authors thank the GO0D MAN partners for their valuable input during the preparation of this paper.

The starting point relates to the observation that Business Process Management (BPM) is perceived as a commodity today. After an evolution from re-engineering and optimization considerations in the 1980s and standardization initiatives thereafter, BPM has now reached the level of an established, industry-independent management standard (Karagiannis and Woitsch 2015). As discussed in Utz and Lee (2017), development efforts along this evolution resulted in a plethora of different process management methods and notations, with BPMN 2.0 (Object Management Group 2011) as the most prominent and widely accepted notation. The standard claims to be an industry-domain independent notation that is understandable by business and technical users alike, therefore bridging the gap of understanding between these two worlds. Looking at past research work and application in the domain, two observations are applicable to the above statement.

Firstly, BPMN 2.0, as the name suggests, is a *notation*, hence neither provides or prescribes analytical processing techniques nor application procedures. The value of the notation in its standard representation lies within the functional capability; models can be transformed from a graphical representation to an executable format in workflow engines. The value of the model is limited to execution and implementation support; verification and validation mechanisms as well as governance and management capabilities are limited and only available through appropriate interpretation and/or mappings during implementation. Rausch et al. (2011) showcase the required transformation on tool level to provide model-value functionality and governance/management structures.

Secondly, the expressiveness of BPMN 2.0 is limited and does not cover all aspects of the manufacturing domain. Work presented by Zor et al. (2011) provides evidence for this observation: either additional elements need to be specified or mappings/interpretations are required to capture the specifics of a production process.

The following chapters aim to overcome these limitations by introducing a hybrid composition of modelling techniques derived from specific sub-domains in the manufacturing field while using a common meta-modelling platform as an invariant. Modelling languages and algorithms for simulation and analysis, summarized as model processing techniques, are realized on abstract level. This enables their application and usage on concrete syntax level of the model structure. The concrete level elements are custom and relate to the type of the manufacturing system and organization-specific requirements. The simulations do not only show how the manufacturing systems should work, but also serve as a means to elicit system requirements (Kaschek et al. 2008). Consequently, the contribution of this research is a procedural framework that guides the design and development of hybrid composed modelling methods. The framework establishes abstract building blocks as the primary artefact steering the design and development.

The remainder of this paper is structured as follows: Section 2 provides details on the idea of Industrial Business Process Management and a review of related work in the area of metamodelling and model processing techniques. Requirements derived from the Agent Oriented Zero Defect Multi-Stage Manufacturing (GO0D MAN) project are presented in Section 3. The core contribution of this paper follows in Section 4, where a procedural framework based on metamodelling building blocks is introduced, comprising a proof-of-concept implementation using the ADOxx metamodelling platform (ADOxx.org 2018). Finally, concluding remarks point to further research directions in Section 5.

## 2 Industrial Business Process Management

The framework presented in this paper is introduced in the context of, and evaluated against the requirements from Industrial Business Process Management (IBPM). Looking at literature from research, the term as such has not been defined yet concretely. Therefore, we propose to position it based on a review of past work and on observations of running research and innovation projects

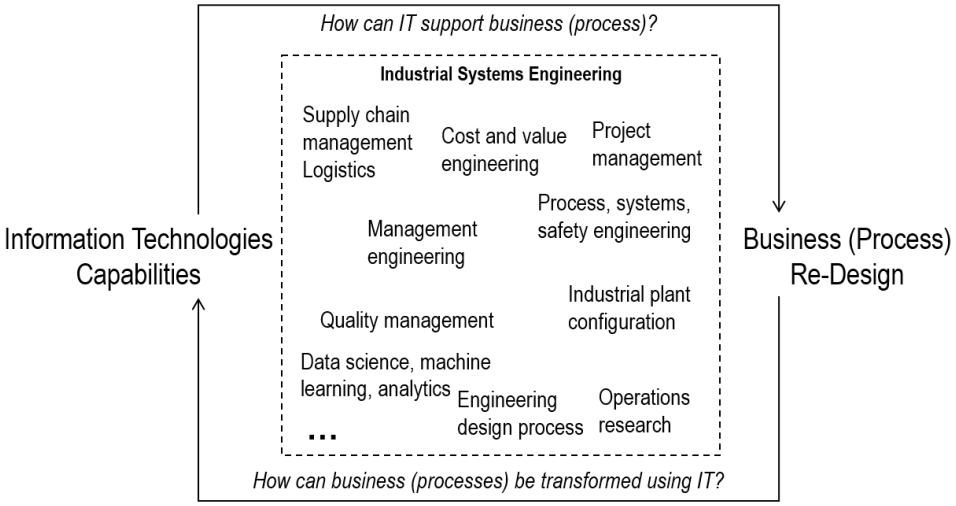



*Figure 1: Industrial Engineering Domains (based on (Davenport, Short et al. 1990) and (Maynard and Zandin 2001))*

in the domain. These projects are focused on the management of industrial manufacturing systems.

A prominent definition from the past contributes to the argumentation in this paper. Davenport, Short et al. (1990) characterize industrial engineering as a combination of "business process redesign with information technologies capabilities". The finding in their publication is that "industrial engineers have never fully exploited" the relationships and capabilities of IT. This statement is regarded as being still valid - even stronger nowadays as the evolution in IT is progressing at a rapid pace. Publications related to bridging the well-known business and IT-alignment gap (Hinkelmann et al. 2016; Hrgovcic et al. 2011; Woitsch et al. 2009) underpin the importance of research in this field.

Figure 1 combines the interactions between business and IT with the disciplines in the domain of industrial engineering. The idea of Industrial Business Process Management builds on the findings that no single, general purpose approach is applicable in such a complex domain. Rather, a hybrid composition of different techniques is required. In the following, first steps towards solving this deficit are proposed by focussing on the support for the design and analysis of complex industrial systems. Industrial engineers shall use an appropriate model-based approach, by utilizing a supporting modelling and model processing environment (Sandkuhl et al. 2018).

## 3 Multi-Stage Zero-Defect Manufacturing in GO0D MAN

The work performed in the context of the Agent Oriented Zero Defect Multi-Stage Manufacturing (GO0D MAN) project[1] defines an application case for Industrial Business Process Management. As an innovation action funded by the European Commission's Horizon 2020 Factories of the Future program, the core idea of GO0D MAN is to integrate and combine process and quality control for a multi-stage manufacturing production (Kimura and Terada 1981).

The collaborative project consists of nine partners from four European countries with different competences and experiences: Industrial end-users from the automotive and white ware production industry provide the industrial requirements for the project and perform the demonstration and evaluation in multi-stage production lines with different levels of automation and production rate. Industrial technology experts in the field of advanced IT solutions (big-data and analytics,

[1] GO0D MAN project page [online]: http://www.go0dman-project.eu/, accessed on January 10, 2018

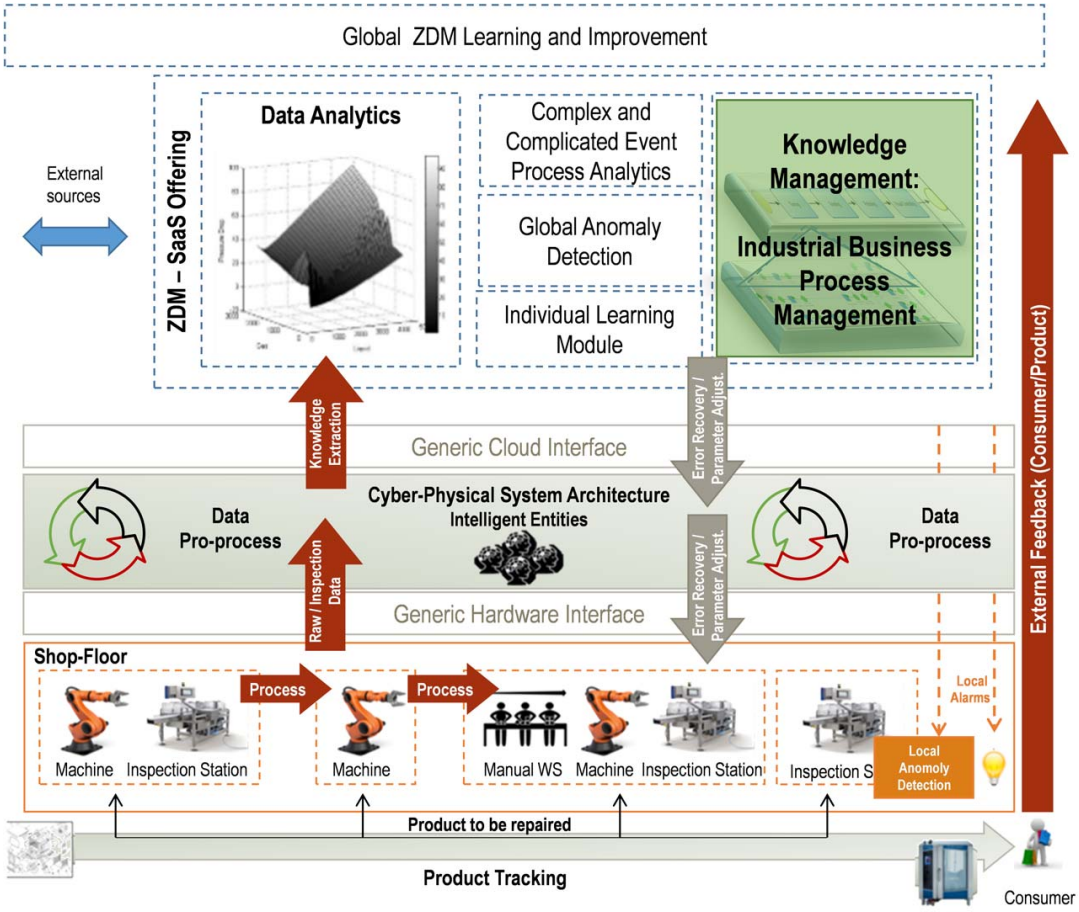



*Figure 2: GO0D MAN Conceptual Architecture (adapted from (GO0D MAN Project 2017c))*

metamodelling and modelling, and automatic systems on quality control) define and develop the software system as depicted in the conceptual architecture described in Section 3.1 and visualized in Figure 2. The project started on October 2016 and will run for 36 months with a budget of 5 million euro.

### 3.1 Conceptual Architecture

The core idea of GO0D MAN is to integrate and combine process and quality control for multi-stage manufacturing production processes into a distributed system architecture built as an agent-based Cyber-Physical Systems (CPS), utilizing smart inspection tools and cloud computing (see Figure 2). Agent technologies, associated to each production stage and product, provide real-time data and defect diagnosis on stage level, inter-stage level, and on global level. All data is stored locally to enable anomaly detection on the edge, and globally to provide complex data processing. The data processing extracts the knowledge and provides it to the Zero Defect Manufacturing Knowledge Management layer.

The Zero Defect Manufacturing (ZDM) strategies as knowledge rules operate on local - directly at the stage - as well as on global system level. Here is where e.g., complex and complicated event process analytics and global anomaly detection is performed. The conceptual models

act as a formalized knowledge base that also contributes, intertwined with the sensor data from the shop floor, toward the identification of defects prior to their occurrence. The utilization of formalized conceptual modelling methods acts as a facilitator by enabling machine processing of the knowledge (Bork and Fill 2014). The ZDM-KM is moreover comprised by an individual learning module that enables an evolutionary learning of past events and the calculation of future events (i.e., defects) at certain stages of the production by inferring live sensor and process data with the existing knowledge base. The specification of the system as a whole is available in (GO0D MAN Project 2017a), further refined as decision rules and strategies in (GO0D MAN Project 2017b).

### 3.2 Simulation of Multi-Stage Production Processes

From a knowledge management and continuous improvement perspective, one objective of the project is to provide an environment for industrial business process management as defined above that enables the analysis and evaluation of hybrid production designs. Hybrid production processes in this case are understood as flexible, loose-coupled combinations of models with domain-specific views and concepts. These views include elements such as a distinction of material flows, information flows, quality data streams, and monitoring techniques, production stages and stations. As detailed in (Utz and Lee 2017), using BPMN 2.0 in such cases has several limitations and drawbacks: the notation lacks domain-specific elements to express different types of flows (i.e., information, material, machine, quality) and quantify them; and, staging logic is missing including specific attributes (e.g., distance, layout, buffer sizes) to emulate the effectiveness of the scheduling.

To overcome these limitations and provide simulation support on the multiple levels of design, the following requirements have been identified:

**Business Process Modelling & Simulation:** The environment should enable the industrial engineer to model and analyse business processes using the BPMN 2.0 notation. BPMN 2.0 should be applicable for any high-level process design and analysis e.g., interaction with suppliers and consumers, sourcing processes, and management system design.

**Block Diagrams Modelling & Simulation:** The detailed production processes should be modelled and analysed using a Block Diagram notation. This notation supports the definition of stages, different flow types, and composition of stations into stages. Thus, employing a higher level of detail and granularity compared to the BPMN 2.0 notation.

**Multi-Stage Path Analysis Simulation:** Based on discrete event simulation algorithms that enable the analysis of the dynamic behaviour of a system, a path analysis algorithm should be selected that can operate on different concrete implementations of the metamodel and provide results for quantitative facets (times, costs), and paths through multi-stage processes as traces.

## 4 Applying Metamodelling Building Blocks in GO0D MAN

In the paper at hand, the framework to realize these requirements is presented, that centres on building blocks. These building blocks are established on three abstraction levels: *Approach*, *Concept*, and *Implementation* (see Figure 3). The three abstraction levels comprise a procedural framework towards the design and development of hybrid modelling and model processing environments. The building blocks enable re-use for an arbitrary number of hybrid methods. Application of the framework is discussed in the context of the GO0D MAN project with the requirements as specified in Section 3.

### 4.1 Approach

On the *approach level*, **abstract metamodel building blocks** are introduced. Each building block on this level couples model constructs with model processing and a description in accordance with the Generic Metamodelling Framework

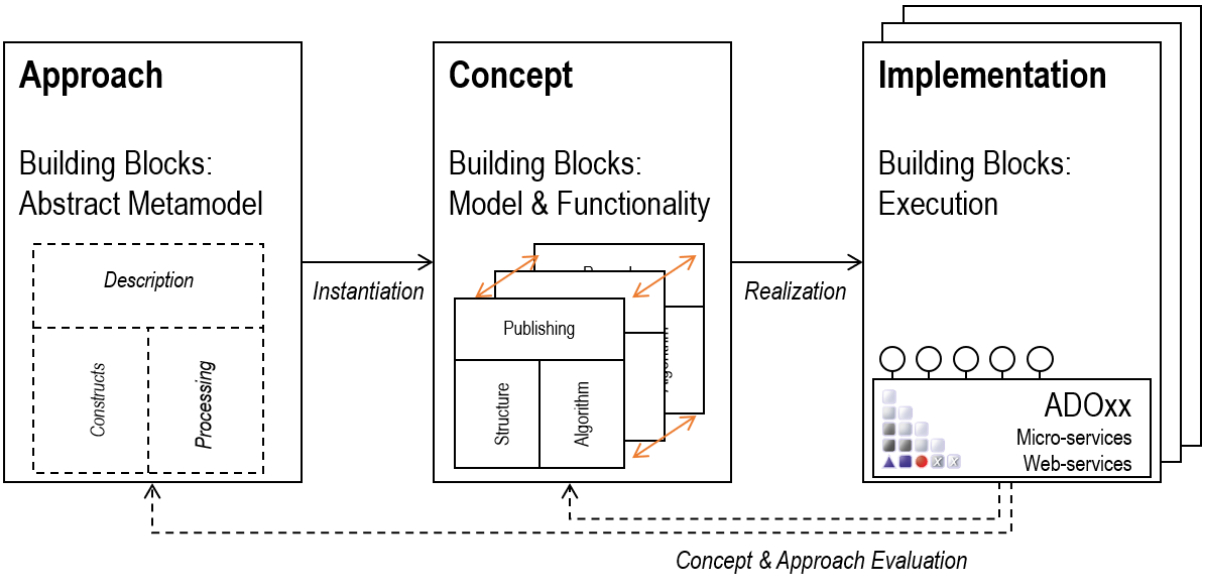



*Figure 3: Metamodelling-based Building Block Framework (Utz 2018)*

defined by Karagiannis and Kühn (2002). Each abstract metamodel building block consists of model **constructs** as the notation/syntax and semantics of the modelling language; the **description** that utilizes the identification and integration of multiple building blocks by means of establishing a modelling procedure; and processing capabilities supporting the modelling procedure and increasing the value of models. This enables the configuration of building blocks for analysis, simulation, and validation/verification algorithms by instantiation of the behaviour on concrete models (being instances of the building blocks on abstract level). Abstract metamodel building blocks comprise the following aspects:

a Constructs: the hierarchy, attributes and properties required for processing are defined on an abstract level. The selection of an abstract block defines the operational semantics of the modelling techniques to be realized.

b Processing: the specification of model processing capabilities, e.g., analysis, simulation and evaluation. The specification stays on an abstract level, utilizing the abstract constructs.

c Description: the description of the abstract metamodel building block that enables the identification and integration of it in more complex scenarios that span multiple building blocks.

An example for abstract metamodel building blocks is discussed in the following. Figure 4 shows the conceptual architecture and how modelling and model processing functionality is provided to the end-user on a role-based level. The common level to derive functionality from is the metamodelling platform and its generic capabilities. It is assumed that these functionalities are common for any kind of implementation and can be used out-of-the-box for implementation.

On top of this layer, the abstract metamodel building blocks are realized. These blocks consist of the model processing techniques and the corresponding metamodel constructs. An example for such an abstract block in Industrial Business Process Management at GO0D MAN relates to discrete event simulation techniques, used to assess process designs using quantitative facets dynamically, such as cycle times, determining bottle-necks in the design, and resource consumption. The related algorithms operate on model constructs that define abstract flow elements (such as activities

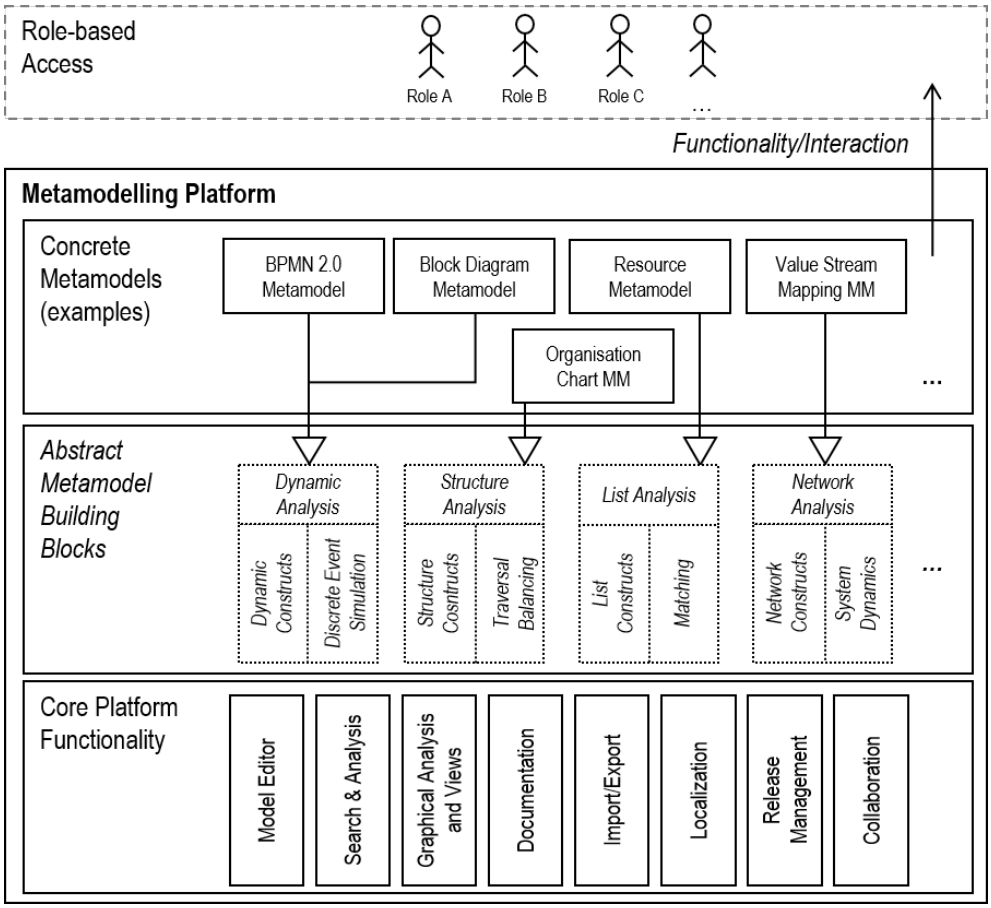



*Figure 4: Using Abstract Metamodel Building Blocks to Design Modelling and Model Processing Environments*

and tasks), logical control elements (such as start events, end events, exclusive/non-exclusive gateways), substantiated with quantitative information such as times and costs. The model constructs can be connected using an abstract logic flow relation including cardinality rules for verification (e.g., number of incoming and outgoing relations for the abstract elements).

## 4.2 Concept

The abstract metamodel building blocks introduced in Section 4.1 are instantiated in the second step of the procedural framework, the *Concept* abstraction level, into **model & functional building blocks**. It is here, where multiple building blocks are combined to realize hybrid modelling and model processing environments, addressing specific requirements of the domain. 'Model & functional building blocks' comprise:

a Structure: the concrete constructs necessary to build an adequate model structure, necessary to create appropriate abstractions of the domain (i. e., models). Existing abstract constructs of multiple metamodel building blocks can be instantiated to realize the necessary concrete structure.

b Algorithm: the modelling procedure by means of orchestrating the model processing functionality provided by multiple metamodel building blocks. Here is, where the abstract processing capability of metamodel building blocks is applied on concrete constructs.

c Publishing: making the model & functional building blocks available for their later utilization in the implementation step.

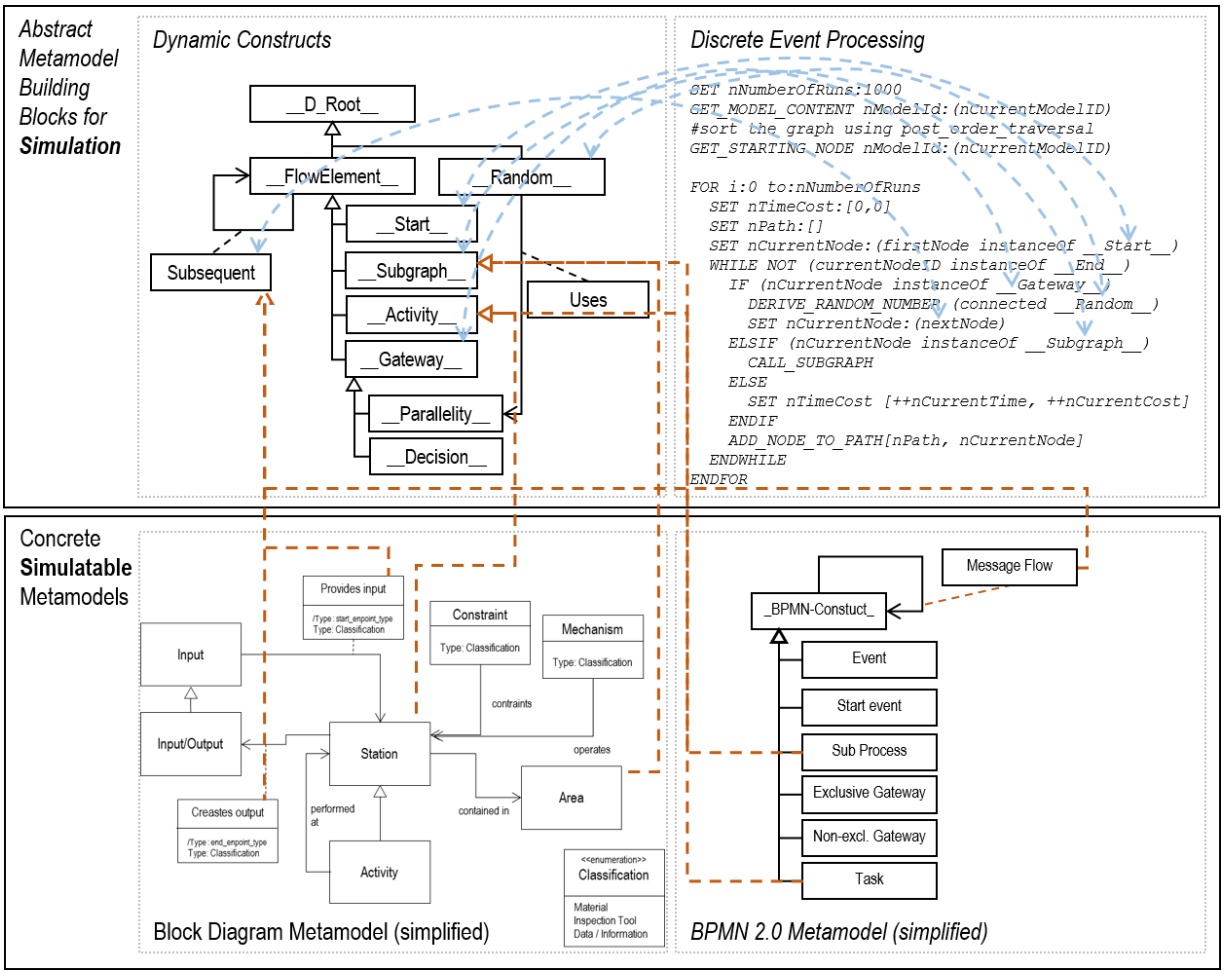



*Figure 5: Building Blocks for Graph-based Simulation of Multi-Stage Production Processes*

On concept level, the abstract metamodel building blocks are instantiated, hereby providing concrete, domain-specific modelling constructs to be used by the designer. In contrast to starting from scratch, the method engineer and modelling tool developer decide on the abstract structure to derive a concrete implementation from, and identify and implement the concrete model constructs. Model processing capabilities are inherited from the abstract implementation and are directly available. It is assumed that such an approach will allow for an efficient implementation, supporting the personalization and customization of platform capabilities and functionality for specific domains. This enables an agile evolution of modelling methods according to (Karagiannis 2015, 2016) without re-implementing functionalities for each update cycle from scratch.

The proposed metamodels and mechanisms/algorithms to support the multi-stage simulation of production processes are graphically visualized in Figure 5 as class diagrams linked to a pseudo code specification of the simulation algorithm. Figure 5 shows how the structure provided in the abstract building block is used by the algorithm.

As the selected path analysis algorithm is generic, it can be applied to any kind of directed graph structure, represented with enriched semantics. The generic realization of the simulation algorithm is visualized by the blue horizontal dashed arrows from the discrete event processing specification to the dynamic constructs in Figure 5. Consequently,

all concrete metamodels derived from this abstract structure are also capable of being simulated. This is visualized in Figure 5 by the orange vertical dashed relationships between the block diagram metamodel, the BPMN 2.0 metamodel and the dynamic structure of the abstract building block.

### 4.3 Proof-Of-Concept Implementation

The 'model & functional building blocks' introduced in Section 4.2 are realized in the third step of the procedural framework, the *Implementation* abstraction level, into **microservice and/or webservice building blocks** that constitute the modelling and model processing environment.

The ADOxx metamodelling platform (ADOxx.org 2018; Efendioglu et al. 2016; Fill and Karagiannis 2013) has been used for the realization of the proof-of-concept. ADOxx uses metamodelling concepts and technologies. It enables the abstract metamodel definition and simulation algorithm and provides a set of platform features out-of-the box without any further implementation need. As an implementation technique, the available extension framework for micro-services has been used.

ADOxx realizes a technical implementation of the metamodel building blocks and fragments that can be dynamically injected into the platform, using a declarative JSON syntax for the structural elements, JavaScript for UI elements, and JavaScript/Java for server side functionality implementation. In this regard, ADOxx acts as an extension and deployment platform. ADOxx supports the efficient composition of hybrid modelling methods by re-using/extending existing implementations. More details on how to combine metamodel fragments can be found in (Živković and Karagiannis 2015). The 'model & functional building blocks' specified in Section 4.2 have been implemented on ADOxx and are provided by means of three micro-services:

1. an abstract metamodel building block consisting of the model structure and the analysis algorithm including result visualisation and mapping;
2. a concrete metamodel building block for Block Diagrams, defining the concrete syntax and notation of the elements used, the diagram container and the user interface elements to trigger the algorithm, and
3. a concrete metamodel building block for BPMN 2.0 in a simplified version, focussing on the core elements, their notation and similar as for 2) the user interaction elements to trigger the simulation algorithm.

The ADOxx micro-services 2) and 3) depend on the abstract implementation in 1). After injection and publishing, the industrial engineer can use the constructs to design the models and trigger the algorithms to evaluate and analyse the models.

Figure 6 shows the user interface of the resulting modelling and model processing environment for multi-stage simulation of industrial business processes. Both concrete metamodels can be used in parallel to define the models, e.g., of a multi-stage oven manufacturing process, and run path analysis algorithms. The figure shows a sample BPMN 2.0 model (top left corner), a sample Block Diagram model (bottom left corner), both for the oven case, and a results window shown after execution of the multi-stage path analysis simulation algorithm (on the right). The generic structure of the building blocks in general, and the simulation algorithm in particular, allow for adaptations and reuse in heterogeneous domains by different modelling languages.

## 5 Conclusion

Based on related works and requirements from industrial research projects, this paper proposed a procedural framework that specifically addresses the requirements of the industrial domain - a hybrid modelling and model processing method. The article showed how this framework can be used to design and develop a modelling and model processing environment based on abstract metamodelling building blocks. Using a multi-stage oven manufacturing example, the utility of the framework has been shown for Industrial Business Process Management (IBPM).

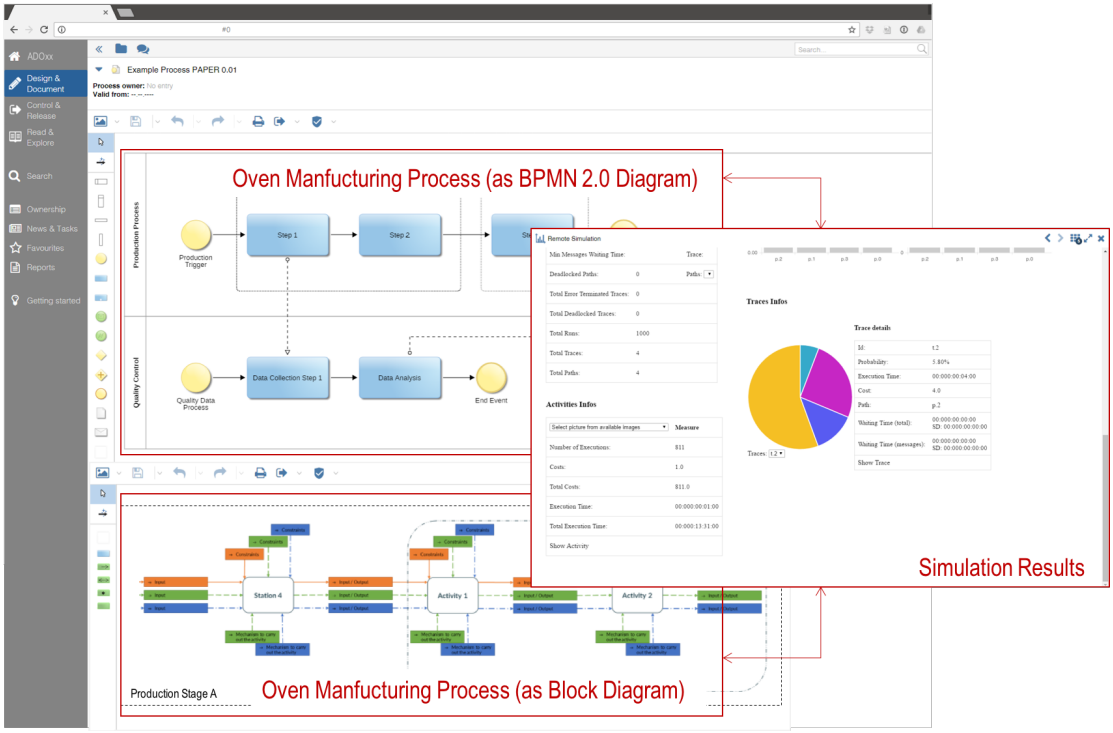



*Figure 6: Graph-based Simulation of Multi-Stage Production Processes with the ADOxx-based Prototype*

As a proof of concept, the metamodelling building blocks have been implemented on the ADOxx platform. Besides the model editor, the tool comes also with a generic implementation of a multi-stage business process path simulation algorithm that can be applied to any concrete graph-based modelling language. The whole implementation relies on abstract metamodelling building blocks that enable efficient adaptation and injection in other hybrid methods. A consideration for abstract metamodelling building blocks framework relates to their composition according to the requirements of the modelling procedure through cross-referencing concrete level implementations and aligning input/output relations along the line.

The article at hand aims to stress the importance of conducting further research in order to empower the industry to overcome the challenges raised by e. g., the digital transformation, enterprise ecosystems, and Industry 4.0. One modelling language and standard modelling languages are not suitable of capturing all upcoming requirements (cf. Pittl and Bork (2017)). It is therefore necessary to think about efficient ways of re-using existing functionality and constructs, and ways of combining them, that copes with the increasing complexity of today's industrial world. This paper proposes abstract metamodelling building blocks as one possible solution for the design and development of hybrid modelling methods and supporting environments.

In our future research, we intend to apply the presented proof-of-concept implementation in the context of the project in order to gain feedback on the maturity of our domain-specific elevation of the industry-independent standard BPMN 2.0 towards a hybrid modelling method that covers the requirements of industrial manufacturing processes management. This combination of multiple modelling languages also comes with risks, e.g., with respect to consistency (Awadid and Nurcan 2017; Bork et al. 2015; Karagiannis et al. 2016a), usability (Sabegh and Recker 2017), and intuitive understanding (Michael and Mayr 2017). We will therefore investigate how industrial engineers can apply our approach in practice in order to

revise the requirements and refine the implementation. The gained feedback will likely also trigger changes to the proposed procedural framework, which will contribute to acceptance and adoption in the future.

## References

ADOxx.org (2018) ADOxx Metamodelling Platform. https://www.adoxx.org/live/home. Last Access: Last Access: January 11, 2018

Awadid A., Nurcan S. (2017) Consistency requirements in business process modeling: a thorough overview. In: Software & Systems Modeling, pp. 1–19

Bork D., Fill H.-G. (2014) Formal Aspects of Enterprise Modeling Methods: A Comparison Framework. In: System Sciences (HICSS), 2014 47th Hawaii International Conference on. IEEE, pp. 3400–3409

Bork D., Buchmann R., Karagiannis D. (2015) Preserving Multi-View Consistency in Diagrammatic Knowledge Representation. In: International Conference on Knowledge Science, Engineering and Management. Springer, pp. 177–182

Cairó Battistutti O., Bork D. (2017) Tacit to explicit knowledge conversion. In: Cognitive Processing 18(4), pp. 461–477

Davenport T. H., Short J. E. et al. (1990) The New Industrial Engineering: Information Technology and Business Process Redesign. In:

Efendioglu N., Woitsch R., Utz W. (2016) A Toolbox Supporting Agile Modelling Method Engineering: ADOxx.org Modelling Method Conceptualization Environment. In: IFIP Working Conference on The Practice of Enterprise Modeling. Springer, pp. 317–325

European Commission (2013) Directorate-General for Research and Innovation, European Factories of the Future Research Association (EFFRA), Factories of the future: multi-annual roadmap for the contractual PPP under Horizon 2020. http://dx.publications.europa.eu/10.2777/29815. Last Access: January 8, 2018

Fill H.-G., Karagiannis D. (2013) On the conceptualisation of modelling methods using the ADOxx meta modelling platform. In: Enterprise Modelling and Information Systems Architectures–International Journal of Conceptual Modeling 8(1), pp. 4–25

GO0D MAN Project (2017a) D1.1 Industrial ZDM Requirements. http://go0dman-project.eu/wp-content/uploads/2017/03/GO0D-MAN-D1.1-Executive-Summary.pdf. Last Access: January 10, 2018

GO0D MAN Project (2017b) D1.3 ZDM Management Strategies and Rules. http://go0dman-project.eu/wp-content/uploads/2016/10/GO0D-MAN-Deliverable-1.3.pdf. Last Access: January 10, 2018

GO0D MAN Project (2017c) Deliverable 1.2 ZDM Management Methodology. http://go0dman-project.eu/wp-content/uploads/2016/10/GO0D-MAN-Deliverable-1.2.pdf. Last Access: January 11, 2018

Hinkelmann K., Gerber A., Karagiannis D., Thoenssen B., Van der Merwe A., Woitsch R. (2016) A new paradigm for the continuous alignment of business and IT: Combining enterprise architecture modelling and enterprise ontology. In: Computers in Industry 79, pp. 77–86

Hrgovcic V., Utz W., Karagiannis D. (2011) Service Modeling: A Model Based Approach for Business and IT Alignment. In: Computer Software and Applications Conference Workshops (COMPSACW), 2011 IEEE 35th Annual. IEEE, pp. 422–427

Karagiannis D. (2015) Agile modeling method engineering. In: Proceedings of the 19th Panhellenic Conference on Informatics. ACM, pp. 5–10

Karagiannis D. (2016) Conceptual Modelling Methods: The AMME Agile Engineering Approach. In: Silaghi G. C., Buchmann R. A., Boja C. (eds.) International Conference on Informatics in Economy. Springer, pp. 3–19

Karagiannis D., Buchmann R. A., Bork D. (2016a) Managing Consistency in Multi-View Enterprise Models: an Approach based on Semantic Queries. In: Twenty-Fourth European Conference on Information Systems (ECIS), ResearchPaper 53

Karagiannis D., Kühn H. (2002) Metamodeling Platforms. In: Bauknecht K., Min Tjoa A., Quirchmayr G. (eds.) Third International Conference EC-Web 2002 – Dexa 2002. Springer, Aix-en-Provence, France, p. 182

Karagiannis D., Mayr H. C., Mylopoulos J. (2016b) Domain-Specific Conceptual Modeling. Springer

Karagiannis D., Woitsch R. (2015) Knowledge Engineering in Business Process Management. In: Handbook on Business Process Management 2. Springer, pp. 623–648

Kaschek R., Kop C., Shekhovtsov V., Mayr H. (2008) Towards Simulation-Based Quality Requirements Elicitation: A Position Paper. In: Springer, pp. 135–140

Kimura O., Terada H. (1981) Design and analysis of Pull System, a method of multi-stage production control. In: The International Journal Of Production Research 19(3), pp. 241–253

Maynard H. B., Zandin K. B. (2001) Maynard's industrial engineering handbook. Sirsi) i9780070411029

Michael J., Mayr H. C. (2017) Intuitive understanding of a modeling language. In: Proceedings of the Australasian Computer Science Week Multiconference. ACM, 35:1–35:10

Object Management Group (2011) Business Process Modeling and Notation Specification 2.0. http://www.omg.org/spec/BPMN/2.0/. Last Access: January 10, 2018

Pittl B., Bork D. (2017) Modeling Digital Enterprise Ecosystems with ArchiMate: A Mobility Provision Case Study. In: International Conference on Serviceology. Springer, pp. 178–189

Rausch T., Kuehn H., Murzek M., Brennan T. (2011) Making BPMN 2.0 Fit for Full Business Use. In: BPMN 2.0 Handbook Volume 2, pp. 189–202

Sabegh M. A. J., Recker J. (2017) Combined Use of Conceptual Models in Practice: An Exploratory Study. In: Journal of Database Management (JDM) 28(2), pp. 56–88

Sandkuhl K., Fill H.-G., Hoppenbrouwers S., Krogstie J., Matthes F., Opdahl A., Schwabe G., Uludag Ö., Winter R. (Jan. 2018) From Expert Discipline to Common Practice: A Vision and Research Agenda for Extending the Reach of Enterprise Modeling. In: Business & Information Systems Engineering https://doi.org/10.1007/s12599-017-0516-y

Utz W. (2018) GO0D MAN Project, D4.2 ZDM Data and Management Environment Implementation. http://go0dman-project.eu/wp-content/uploads/2018/05/Abstract-D4.2-ZDM-Data-and-Management-Environment-Implementation.pdf. Last Access: to be published

Utz W., Lee M. (2017) Industrial Business Process Management Using Adonis Towards a Modular Business Process Modelling Method for Zero-Defect-Manufacturing. In: Industrial Engineering, Management Science and Application (ICIMSA), 2017 International Conference on. IEEE, pp. 1–5

Woitsch R., Karagiannis D., Plexousakis D., Hinkelmann K. (2009) Business and IT alignment: the IT-Socket. In: e & i Elektrotechnik und Informationstechnik 126(7), pp. 308–321

Živković S., Karagiannis D. (2015) Towards metamodelling-in-the-large: Interface-based composition for modular metamodel development. In: International Conference on Enterprise, Business-Process and Information Systems Modeling. Springer, pp. 413–428

Zor S., Schumm D., Leymann F. (2011) A Proposal of BPMN Extensions for the Manufacturing Domain. In: Proceedings of the 44th CIRP International Conference on Manufacturing Systems

# The HBMS Story

## Past and Future of an Active Assistance Approach

Judith Michael[*,a], Claudia Steinberger[a], Vladimir A. Shekhovtsov[a], Fadi Al Machot[b], Suneth Ranasinghe[a], Gert Morak[a]

[a] Institute for Applied Informatics, Alpen-Adria-Universität Klagenfurt, Austria
[b] Leibniz Center for Medicine and Biosciences, Research Center Borstel, Germany

Abstract. *The aim of the Human Behavior Monitoring and Support (HBMS) project has been to actively assist individuals in activities of daily living and other situations using users' own episodic knowledge. This knowledge is represented and preserved in HBMS in the HCM, the Human Cognitive Model, expressed in the domain specific modelling language HCM-L. HCM also forms the base for reasoning, model matching and support state visualization. Moreover, in the HBMS-System conceptual models are also used to define interfaces to activity recognition systems, support clients and data available in the Semantic Web. Thus, we see HBMS-System as an application of the Model Centered Architecture (MCA) paradigm. This paper describes how the project evolved over time, its main challenges and milestones, its main processes and their dependencies, and what is going to happen in the next future.*

Keywords. Conceptual Modeling • Active Assistance • Human Behavior Monitoring and Support • Model Centered Architecture

## 1 Introduction

**Motivation.** This paper tells the story of a project, in which the authors have been working over the last few years. The motivation for this project can be explained best by retelling a story a researcher and later HBMS project manager told as his vision at the very project beginning:

*Imagine your mother suffering from incipient dementia. As time goes by, she loses her episodic memory more and more. For example, she forgets how to use her video recorder to watch beloved music videos and begs you to explain it to her. You are going to show it to her repeatedly and she can do it for a while, but then she will forget it again. Forgetting happens faster and faster. After some more time, you realize she does not ask you anymore to explain it to her. She does not want to burden you. If you try to explain it to her again, she tells you to stop, because she knows she won't remember. She stops watching her beloved videos at all. And you know she also stops doing other things she likes, because she can't remember how to do. In this way her autonomy decreases very quickly. Wouldn't it be great if she had a permanent electronic assistant to help her to accomplish her tasks of daily living and to support her to live autonomously as long as possible?*

The researcher with this vision was Heinrich C. Mayr and the idea was transformed into the Human Behavior Monitoring and Support (HBMS) project, funded by the Klaus Tschira Stiftung gGmbH in two project parts from March 2011 to January

* Corresponding author.
E-mail. judith.michael@aau.at
We thank the Klaus Tschira Stiftung GmbH for their project funding and all colleagues and students involved in the development of the HBMS project.
Note: Although Heinrich C. Mayr is not an author of this article, the ideas and work presented in this article were developed together with him. (Mayr et al. 2016)

2016 with a funding amount of approx. 1.000.000 Euro.

**Research idea.** The idea of HBMS was to develop a system which provides ambient assistance to support the autonomy of a person in case of a decreasing memory. This is to be achieved through the conceptualization of the person's episodic memory, the establishment of the model of this knowledge and the use of this model for support. Giving support means to help people to remember how they performed a certain activity by reactivating already existing memory anchors. A memory anchor is a simple stimulus that influences the state of mind. By providing retrieval cues, it is easier to remember experienced situations (cued recall) (see Strube 1996, p. 204 and p.196). Thus, the person is supported by the HBMS-System with his or her own behaviour knowledge by providing retrieval cues.

**The aim.**The HBMS project aims at deriving support services from integrated models of abilities, current context and episodic knowledge that an individual had or has, but has temporarily forgotten. The core of the HBMS-System is the Human Cognitive Model (HCM) which preserves the episodic memory (Tulving 1972) of a person as conceptual models of behaviour linked to context information related to these activities (see Figure 1). The acronym HCM was not coincidently chosen - it was named after the nickname of Heinrich C. Mayr (derived from his initials).

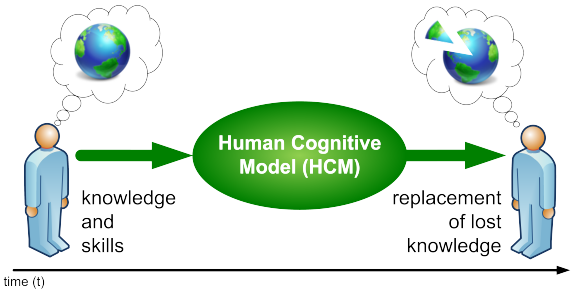



*Figure 1: The main HBMS idea*

**How to meet the aim.** The developed system should be able to monitor actions, identify activities, create conceptual models out of these activities, transform them into an ontology representation which forms the knowledge base, draw conclusions out of this knowledge base, and use this knowledge to provide support. Thus, the HBMS team has created an active assistance (AA) system, which realized these aims step-by-step.

**Outline.** First, an overview of the project is presented in section 2, sketching our key activities during the project, which will be examined more closely in the subsequent sections. Thus section 3 presents our approach for end user involvement and performed evaluations. The whole HBMS project was accompanied by a continuous analysis of the state-of the-art of all relevant areas, which is presented in section 4. In section 5 our Domain Specific Modelling Method including our modelling language HCM-L and its chronological development is treated. The HBMS-System development is presented in section 6: The HCM-L Modeller (Section 6.1), observation aspects such as the HBMS observation interface, an alternative to real human observations via a simulator component (Section 6.2), solutions to handle business intelligence like reasoning approaches and a real-time visualization for behaviour models (Section 6.3), as well as different user interfaces (Section 6.4) complete this section. The last section summarizes the projects' impact and discusses future ideas.

## 2 Past and Future of HBMS

This chapter aims to sketch what happened in HBMS previously and what else we have planned. As conceptual modellers, we present this of course in form of a conceptual model: the development of HBMS, the processes we were involved in, the deliverables and responsibilities. We have documented what has happened in HBMS until now and what is going to happen in the next future.

Figure 2 conceptualizes our top level research processes and structures the work which has been done so far. To keep Figure 2 as simple as possible we modified BPMN slightly for our purposes:

- Our HBMS process model is both prescriptive and descriptive. Thus we have added a timeline showing past, present and future activities.
- To represent the academic impact of HBMS clearly, we have added our deliverables published in the course of time to our process model as data outputs on the timeline.
- All activities in Figure 2 can be seen as sub-processes. The sub-processes are detailed in more depth in the following subsections of this paper.
- The most important interactions between sub-processes of different pools are sketched using message flows.

Figure 2 comprises the following pools:

**Evaluation.** It has been a core part of HBMS to analyse user requirements and to evaluate the practicability and the benefits of HBMS throughout the entire project. User centred requirement analysis workshops were mainly conducted with elderlies in small groups of up to 25 persons. Surveys were conducted among participants of all ages to obtain feedback on design and user interface approaches. Practicability evaluations were done with volunteers mainly in user experience labs, which were organized primarily as part of 'long night of research' (LNF) events, which attracted thousands of people to our University. Evaluation processes were influenced by other processes and influenced processes in other pools throughout the entire project, especially to a high degree during the first third of the project period.

**Analysis.** During our analysis process we investigated the current state of the art in the domains of active and assistive systems, modelling languages, context and process ontologies, activity recognition approaches, and observation technologies. This pool also hosts processes, which define requirement specifications as a basis for processes in other pools. Inputs came mainly from evaluation processes as mentioned above.

**Domain Specific Modelling Method (DSMM) Design.** HBMS aims to assist individuals in their activities of daily living and other situations using their own episodic knowledge as well as knowledge about their context including the resources users have at hand. This knowledge is preserved in HBMS as a Human Cognitive Model (HCM), which contains behavioural and contextual elements. We have worked on the domain specific modelling language, called Human Cognitive Modelling Language (HCM-L), and developed several incremental versions over time.

This work was mainly influenced by analysis and evaluation processes and affected essentially the development process of the modelling tool (HCM-L Modeller).

Today, the actual version of HCM-L comprises really powerful context modelling elements for user profile modelling, resource modelling and environment modelling.

The development of the HBMS-System included extensive research activities to create the HCM-L Modeller, to develop appropriate user interfaces, to handle user observations and to sharpen the HBMS-Systems' intelligence. The architecture of the HBMS-System corresponds now to the Model Centred Architecture (MCA) paradigm, which was proposed by Heinrich C. Mayr in (Mayr et al. 2017). Details regarding the HBMS-system architecture are available in section 6.

**HCM-L Modelling Tool.** The HCM-L Modeller has been developed as a modelling tool that supports HCM-L to enable computer aided generation, integration, modification and management of behaviour and context models. A basic version of the HCM-L Modeller is available via the Open Models Initiative for download.

**User Interface.** HBMS user clients were intended to help users interactively and unobtrusively in performing their activities. During project progress the clients and their user interfaces have developed from rigid, mainly text oriented web-applications to personalised multimodal apps running on various mobile devices. Today the communication between the HBMS-System and the user clients works unidirectionally and environmental sensors are required to feedback user

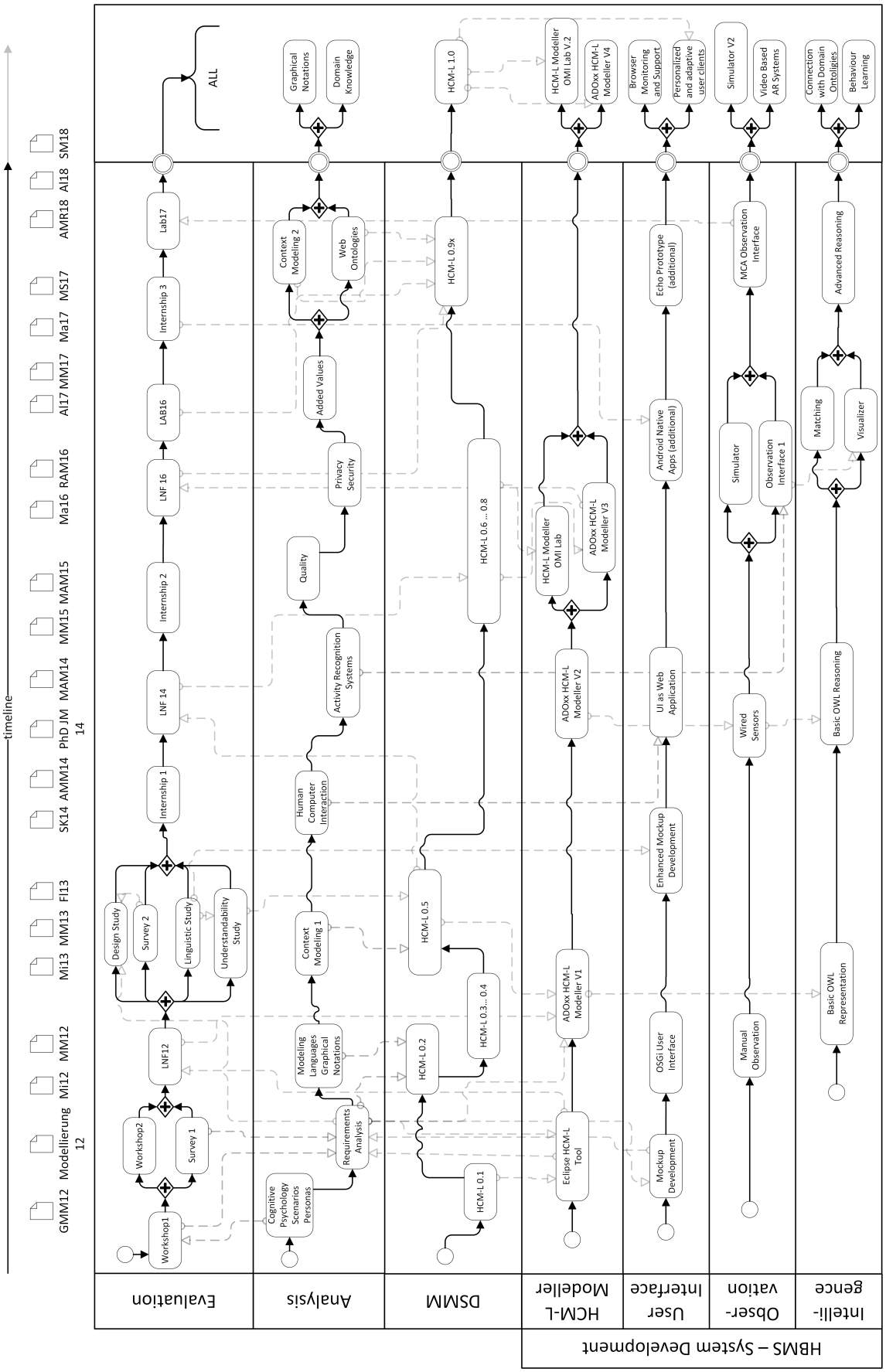



*Figure 2: Past and Future Model of the HBMS lifetime*

behaviour into the support system again. However, now we are going to expand the user client functionality in order to feedback user input in a bi-directional form e. g. via voice assistance to bridge the current activity recognition gap (Steinberger and Michael 2018).

**Observation.** Observation processes dealt with possibilities to monitor a user and to infer and recognize inhabitants' activities, changes or anomalies, continuously in real time in a progressive way based on activity models. We set up different labs using various networked sensor technologies and context management platforms and even developed an activity recognition interface to integrate latter. In the beginning of HBMS, we had high expectations regarding current activity recognition capabilities, but we had to realize at the end that current activity recognition approaches are too coarse to monitor all needed details for HBMS. As explained above, now we are going to add a user client to the HBMS-System which interacts with the user adaptively and in a smart way – to find out if a proposed activity has been done (as an alternative user activity recognition service) – and communicates the answer back to the cognitive assistance system. Intelligence. The HBMS-System increased its level of intelligence step-by-step by developing reasoning, matching and visualisation mechanisms. The first HBMS-System prototype included basic reasoning mechanisms within the HCM-L Modeller. The reasoning approaches to improve activity recognition were tested on laboratory datasets. The latest prototype covers the matching of recognized actions with operations in the knowledge base as well as a direct visualization of the matching results for end users.

The next sections will focus on the processes described above in more detail.

## 3 User Involvement and Evaluation

HBMS has been a user centred project and user involvement has been an integral part of the project from the very project beginning. The goal of our user evaluations was to derive system requirements from the user's perspectives mainly in the field of acceptance, interfaces, usability, interaction modes and model comprehensibility.

The evaluation pool in Figure 2 shows a rough overview what has been done for evaluation purposes:

During the first phase of the project, a particular attention has been paid to user involvement. Different types of user evaluations validated existing concepts and provided valuable input for requirements analysis. In a first **workshop**, we discussed with a group of about 40 senior students (attending the 'Senior Studium Liberale') their acceptance of AAL systems, possibilities for assistance and their ideas for user interfaces. In the second workshop, the group of senior students was extended by younger students and their desired assistance situations were discussed (see Figure 3).

*Figure 3: Discussions in the second Workshop*

Simultaneously an **empirical study (online survey)** with 203 persons of all ages was conducted on opportunities and obstacles of active assisted living (Mi13). By this way, the first HBMS prototype was strongly influenced by prospective users.

The *'Long Night of Research' (LNF)*, a biennial **public event** in Austria, where numerous national research institutions open their doors to the public, was repeatedly an important stage for us to present our intermediate results and get feedback on it. LNF has been welcomed by the public very well and the University attracted approx. 7000 visitors at each event. The scenarios used during our evaluations originated not only from the area of

'activities of daily living' (Katz 1983; Lawton and Brody 1969) but also from more technical ones (see Figure 4). LNF user feedbacks always enriched the subsequent project steps.

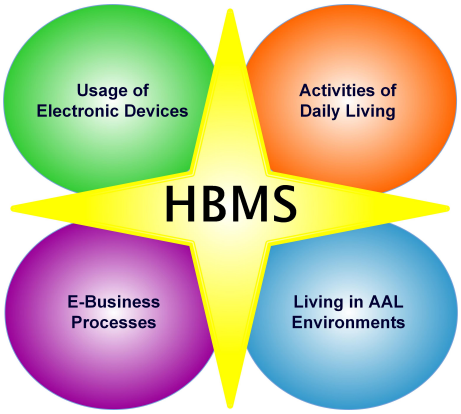



*Figure 4: Project relevant scenarios*

In 2012 *(LNF12)* the first version of HCML-tool as well as a first support client were presented to approx. 50 volunteers. User's coffee preparation behaviour in cooperation with an automatic coffee machine was modelled, these models were discussed and different support possibilities were checked.

In 2014 *(LNF14)* an advanced prototype was presented to the public. HBMS had learned to sense some context information during the last 2 years and was presented in a first small experimental lab to approx. 50 volunteers. User interfaces on different clients were used to support some activities of daily living. The HCM-L Modeller was still a rapid prototype for analysis purposes integrating modelling and activity recognition capabilities.

In 2016 *(LNF16)* the HBMS-System had been grown up and was presented again in a particular lab environment where activities of daily living, the handling of electronical devices and online transactions were supported. Additionally, behaviour model visualizations and priorities for specific user clients were evaluated with approx. 100 heterogeneous volunteers.

As this lab environment had turned out to be very popular we evaluated system improvements based on LNF16 feedbacks a few month later again with 50 high school students.

To enhance the support client presented in LNF12, a **user design study** was performed, testing different user mock-ups in nine scenarios. The testing was done one at a time with 55 persons of all ages, gender and education levels and brought new insights into user surfaces and ways of user navigation and interaction (Strobl and Katzian 2014). Simultaneously an **empirical study (online survey)** was conducted to collect user client design ideas from 338 persons of all ages. The survey showed a high demand of autonomy of the end users, which means that they want to decide when to get support and how.

As support systems are supposed to interact with the user in a clear way and user manuals normally are not well suited to be used online or orally, linguistically reduced user instructions were tested in a **qualitative study** with app. 50 persons (Fliedl et al. 2013).

Moreover, a **qualitative study** including a qualitative content analysis (Mayring 2010) was carried out with 54 non-technology students to evaluate the intuitive understandability of HCM-L models and the notations of HCM-L concepts. (Michael and Mayr 2017)

Since 2013, we annually employed up to 6 interns for *evaluation purposes* during summer time. The group of internship 1 tested the tools, modelled different scenarios with the HCM-L Modeller and discussed their resulting models at home with friends and their relatives to investigate its comprehensibility. The group in internship 2 had to lay an eye on existing models and test the user client's surface, interaction and navigation capabilities. The group of internship 3 focused on possibilities for user modelling and robots as potential user clients.

Now, we are currently working to adapt the HBMS observation interface by integrating it with a wide variety of the existing activity recognition systems with traditional sensor based input and output formats. Evaluations to test its adaptivity and flexibility with the HBMS-System are currently running. Moreover, evaluations to test the

observation interface with video based activity recognition systems have been planned for the next future.

We are also working on the evaluation of various notations for modelling the advanced HBMS context, namely user models, environment models and spatial models.

## 4 Analysis

The entire HBMS Project was accompanied by a continuous analysis of the *state-of-the-art of assistance systems*. *Projects* in the ambient assistance domain as well as *current research*, published at conferences and in journals, were systematically analysed according to their type of assistance. *Standards* in the AAL domain were evaluated and ongoing projects related to standards were pursued. Some of the topics, which were also of great interest to us, are listed below:

**Cognitive Psychology.** In an extensive analysis compendium, we investigated factors for the reduction of individual memory capacities and cognitive limits (e. g. stress, sleep deficits or depressions as age independent factors, as well as age related factors).

**Scenarios.** We analysed areas of assistance systems and identified the most important application scenarios of electronic devices, e- processes and everyday activities for our research, e. g. (Bonfiglio et al. 2008).

**Personas.** Personas are prototypical persons, who could be seen as future users of a system. Based on an analysis of the CURE Elderly Personas (Wöckl et al. 2011), we have identified relevant user groups for the HBMS-System.

**Requirements Analysis.** We identified requirements for our future modelling tool and investigated different technical solutions. (see Section 6.1)

**Modelling Languages.** As conceptual modelling played an important role for our project, we investigated which existing modelling languages were able to represent human activities and which language could be used as a base language for our DSMM to be developed.

**Graphical Notations.** Approaches for the intuitive understanding of modelling languages and graphical notations were investigated.

**Context Modelling.** We conducted a comprehensive analysis of context modelling approaches used in different domains. Whereas the first findings from (Kofod-Petersen and Cassens 2006) resulted in a general context structure focussing on behaviour (Michael and Mayr 2013), further work discussed the structural elements in detail (Michael and Steinberger 2017).

**Knowledge Representation.** We investigated different approaches for knowledge representation with a special focus on ontologies, semantic languages and related tools. We analysed different web ontologies for their usefulness to include general and domain knowledge into the HBMS knowledge base. Activity Recognition. Various systems for activity recognition were evaluated and categorized based on their recognition methodology. (Ranasinghe et al. 2016)

**Human Computer Interaction.** An important aspect for the development of user support clients in HBMS were useful technologies for human-computer interaction. Thus, speech recognition and synthesis, humanoid robots, and other technological possibilities were evaluated.

**Quality.** We analysed approaches for quality understanding, e. g. (Lindland et al. 1994; Schütte 1998), and defined useful quality categories for the HBMS project and techniques to reach a set of quality requirements.

As the project continued to make progress, we started to evaluate *'added values'*: In the initial phase of the HBMS-System, when the system needs to learn the behaviour of a person, no direct benefit is generated. Thus, we have evaluated existing products on the market on their compatibility with the HBMS-System and usefulness for end users.

Other important aspects for a real life usage of the HBMS-System were confidentiality, integrity and authenticity considerations. Thus, we evaluated the system and users' wishes and developed a *security and privacy concept*.

## 5 Domain Specific Modelling Method

It was clear from the very beginning, that conceptual models should build the core knowledge base of the HBMS-System. Thus, a Domain Specific Modelling Method (DSMM) including a modelling language called Human Cognitive Modelling Language (HCM-L) was created and further developed. The first stable version was presented in the dissertation "Cognitive Modelling for Assistance Systems" (based on Michael and Mayr 2013).

The first version of the HCM-L meta-model included operations, things and conditions as model elements and, as further specializations, different variants of operations and conditions as well as some sort of branching and merging concepts (called gateways). The modelling process was realized in a first prototype of the Modeller. Nevertheless, first evaluations showed us that several aspects were still missing.

Our former experiences in conceptual modelling with *KCPM (Klagenfurt Conceptual Predesign Model)*, a lean, user-centred language for software requirements modelling (Kop and Mayr 1998), helped us to carry out further improvements. Consequently, the KCPM concepts were evolved and adapted to cognitive modelling. Therefore, we could benefit from adopting from the KCPM approach the thing concept (a more intuitive way of abstracting from classes and attributes), as well as the approach for relating resources and actions.

Based on several project discussions, the second version of the meta-model was developed: It still focused on behavioural aspects and included an aggregation of several operations into an activity (called *Behavioural Unit, BU*), calling-, participating- and executing-connections between operations and things, person and location things as specializations of thing and different variants of connections like is-a, part-of and property connections. Obviously, this version (0.2) was hardly related to version 0.1. Most behavioural aspects remained stable up to the next versions.

HCM-L version 0.2 was published in (Mayr and Michael 2012), where we presented a semantic analysis based on control flow workflow patterns (van der Aalst et al. 2003). We showed that HCM-L was – for the domain of Ambient Assistance – sufficiently powerful and met the requirements for modelling human daily activities better than general purpose languages like BPMN or UML (see e. g. Wohed et al. 2005).

Figure 5 shows the excerpt of a BU model including two operations 'take a cup from the cupboard' and 'put the cup on the drip tray' as well as related things: the person 'Francis' as calling and executing thing and the 'cupboard' and 'coffee cup' as participating things.

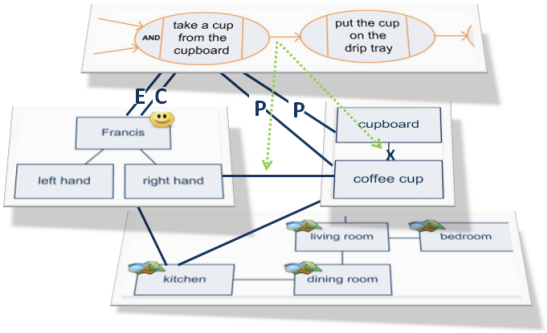



*Figure 5: One step of the behaviour context in detail*

Literature research in cognitive science and psychology took us to *activity theory* (Leont'ev 1978), which observed the nature of human activities on three levels: The level of the *activity* (the overall process), the level of the *action* (subtasks) and the level of *operations* that realize actions. While activities are driven by motives, each individual action pursues a specific goal and operations are related with instrumental conditions and constraints. Our HCM-L meta-model already included these levels but the essential concept of 'goal' was missing until HCM-L version 0.3.

We decided to move conditions into operations and realized that the concept flow, a connection between two operations, had to be added (version 0.4). One development stage further, we added some attributes to goals, operations and BUs and a relation between operations and properties of things. This version 0.5 of the HCM-L meta-model was published in (Michael and Mayr 2013).

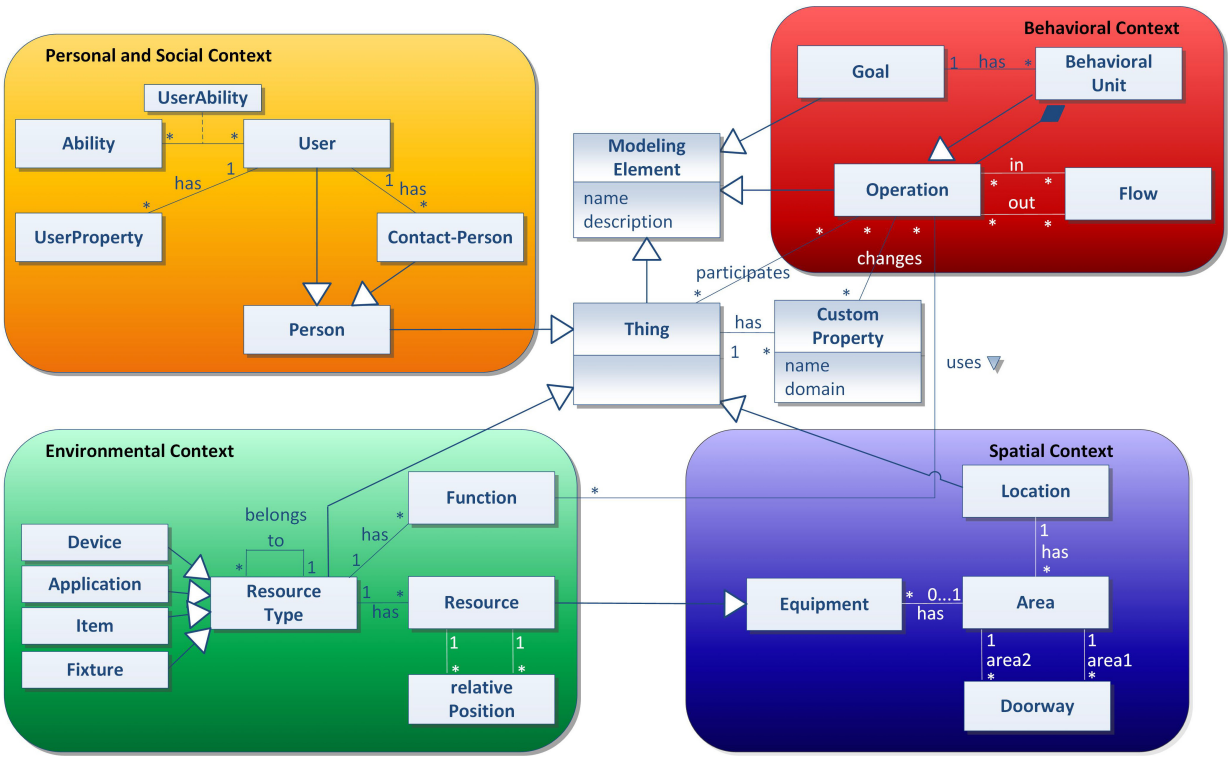



*Figure 6: HCM-L Meta-Model v 0.93*

The next changes were minor ones: some attributes were deleted, attributes of goals were move to the relation between operations and BU, a relation between goal and thing was added and a variant of things called service thing was added.

The next essential HCM-L enhancements were published in (Michael and Steinberger 2017), where the concepts regarding the structural context characteristics were defined in more detail. Accordingly, it was possible to model for example abilities of users (personal and social context), functions of different resources, different variants of resource types (environmental context) and equipment in different areas (spatial context). Figure 6 shows the current state of the HCM-L meta-model including the different context clusters (based on Kofod-Petersen and Cassens 2006).

One of the next HCM-L development steps was related to an approach to integrate semantic annotated user manuals into the HBMS-System: schema.org (Guha et al. 2016) was applied to semantically annotate such manuals. Corresponding mark-up data was used to import domain knowledge and "how to use"-resources, into the HBMS-System. Thus, the meta-model had to be partly adapted concerning instructions, warnings and problem situations (Steinberger and Michael 2018).

Next important HCM-L additions are in progress. They are related to *behaviour goals*: The concept goal needs to be investigated in more detail to be able to express, e. g., relations between different goals, hierarchies, and relations to domain and fundamental ontologies as a compensation for the semantic memory (general knowledge about the world and its' concepts).

Together with the HCM-L meta-model also our graphical notation changed. The elements were revised in several iterations based on evaluation feedbacks and because of findings from (Moody 2009), e. g., icons were added, the colour and thickness of different connections were changed for a higher visual distance and new graphical

elements were added accordingly to additions in the HCM-L meta-model.

To make it clear, how to use our DSML systematically for creating models, we described the modelling procedure (Michael et al. 2015) based on the work of Karagiannis and Kühn (2002). It treated e. g., with which diagram type the modelling process should start or what aspects were to be modelled first.

Additionally, we have taken our experiences on creating a DSMM on a more general level (Michael and Mayr 2015) by describing the way of how to create a modelling method for ambient assistance, based on the work of Frank (2013).

In parallel to HCM-L development, we created a HCM-L modelling tool, which enabled us to work in practice with the DSML (see section 6.1).

## 6 HBMS-System Development

The development of the HBMS-System took place in *several stages*, whereby the prototypes were regularly evaluated (see section 3). This section sketches the most important HBMS-System versions from the first prototype architecture to the final model centred architecture of HBMS-System and particularly emphasizes the development of the HCM-L Modeller, the observation capabilities of HBMS-System and the intelligence of HBMS-System during the development process.

**HBMS-System Version 1- Eclipse and OSGI**

The focus in the first HBMS development phase was on the following aspects:

- Tool support in HCM-L modelling.
- Mapping of HCM-L models to a suitable knowledge representation for future reasoning purposes.
- First HCM knowledge base with suitable access interfaces
- First user support clients.

Figure 7 shows the architecture of the first HBMS (rapid) prototype evaluated at LNF 2012.

It included an Eclipse-based modelling tool, an ontology and a server and dialog component. The prototype realized the following functionalities: modelling the user behaviour by means of the first HCM-L version, transformation of these models into OWL-Lite (using JENA framework), storing the results in a filesystem, manual upload of the OWL files into a service domain, and user support via first mobile user clients.

**HBMS-System Version 2 - ADOxx**

Based on our experiences and evaluation results the next significant HBMS-prototype changed its paradigm. The next significant version of the HCM-L Modeller was based on the meta-modelling platform ADOxx (Fill and Karagiannis 2013), reasoning mechanisms were included directly in the HCM-L Modeller, and the user support client was extended to a web-based version. This prototype included also the following new functions: construction of more complex behaviour models according to the next HCM-L version, behaviour animation, embedding of media data, model export and import interfaces, transformation of models to OWL DL, and a first simple observation functionality using wired sensors. This prototype was very helpful to analyse and evaluate our requirements but was too monolithic and inflexible in its architecture.

**HBMS-System Version 3 - MCA**

The next prototype of the HBMS-System (which was presented and evaluated at the LNF16) was implemented according to the *Model Centred Architecture (MCA)* paradigm (see Figure 8). MCA was proposed in (Mayr et al. 2017) as a generalization of our experiences from several projects, including HBMS. According to the MCA paradigm, we understand information system development processes as mere modelling processes and focus on models (formed with the means of DSML) at any development stage up to the running system. The models are taken as the core of a system for both the application functionality and the system's interfaces.

According to (Mayr et al. 2017, Shekhovtsov et al. 2018), the MCA paradigm is defined on the following three levels:

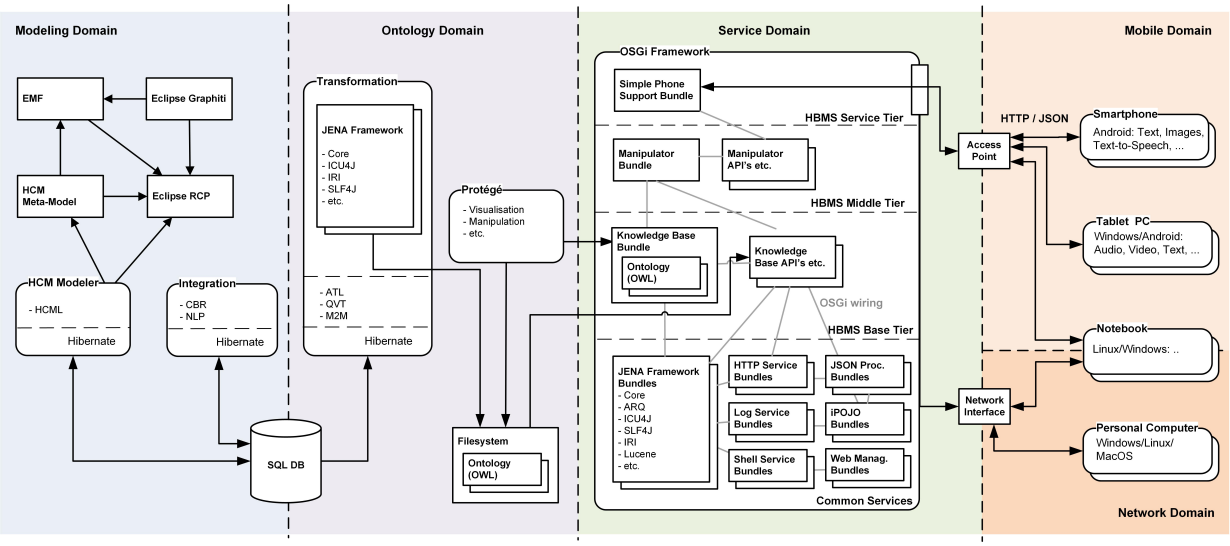



*Figure 7: Architecture 2012 (adapted from (Michael et al. 2013))*

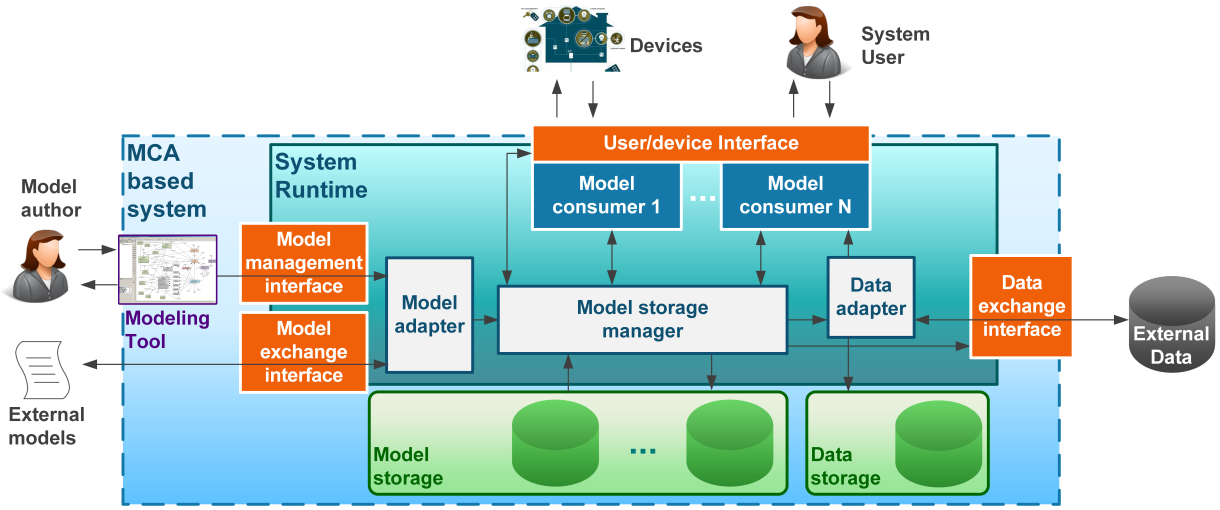



*Figure 8: MCA patterns (adapted from (Mayr et al. 2017))*

1. as a meta-model hierarchy on MOF levels: M2 (the DSML meta-model definition based on a meta-meta-model), M1 (the specification of the system components and its data) and M0 (models of concrete objects, functions and processes); we call this definition a *semantic core* (Object Management Group OMG: n.d.);
2. as a language specification hierarchy representing the core semantics using different syntactic constructs of the appropriate representation languages. The hierarchy descends from grammar definitions, to concrete grammars for the languages representing semantic concepts, to concrete language representations of those concepts. This concrete language representations can even refer to constructs from different levels, which means the concrete languages can be used to represent instances, models, meta-models, and meta-meta-models, even together in the same project;
3. as a set of architectural patterns including the *Modelling Tool* pattern (components used for

creating models), the *Model Transfer Interface* and the *Model Adapter* patterns (for the components providing the models to the consumer system), and the *Model Consumer* pattern which describes the components which use the models to provide the functionality of the MCA-based solution. It also includes the *Device and User Interface* patterns which are the model-based interfaces to the model consumers and the *Data Storage* pattern (see Figure 9).

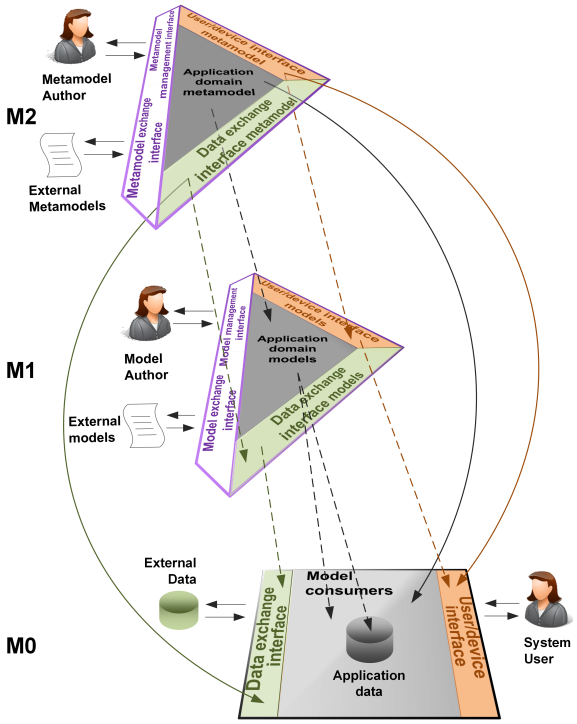



*Figure 9: Model Centered Architecture (adapted from (Mayr et al. 2017))*

In version 3, the HBMS-System was redefined to comply with MCA. The corresponding architecture is depicted on Figure 10.

According to this architecture, the HBMS-System includes the following components:

1. **HCM-L Modeller** is used for creating and maintaining HCM-L models and the transformation of the created models to the HBMS kernel using the HBMS model transfer interface.
2. **HBMS Observation Interface (HBMS-OI)** includes an activity recognition system adapter (HBMS ARS adapter). Together, they form a middleware listening to inputs coming from the activity recognition system and making it HBMS-compliant; these components implement the Device Interface MCA pattern.
3. HBMS Kernel is the central component of the system. It is accessible via the set of kernel interfaces and contains the following internal components (all implementing Model Consumer MCA pattern):
   a) *HBMS Observation Engine* responsible for communicating to the Activity Recognition System (ARS) through HBMS-OI;
   b) *HBMS Behaviour Engine* responsible for handling the behaviour data arriving from the HBMS Observation Engine in context of the current HCM;
   c) *HBMS Support Engine* responsible for controlling the behaviour of the assisted users;
   d) *HBMS Data Management Subsystem* responsible for managing HBMS data;
   e) *HBMS knowledge base* which stores the current (HBMS situational cache) and historical data (HBMS case base) together with its description (HCM storage) in a knowledge-oriented format;
4. **HBMS kernel clients** (monitoring client, simulator client and care worker client) to support monitoring and administrative tasks based on the information provided by the kernel.
5. **HBMS support clients** which implement multi-modal interfaces for end user support.

For evaluation purposes, it is possible to install and run all HBMS-System components on one computer as 'HBMS-System in a box' and to use the 'HBMS Simulator' to 'emulate the user behaviour in a smart lab' virtually (see Section 6.2).

The next sections describe the development stages of the HCM-L Modeller (section 6.1), the

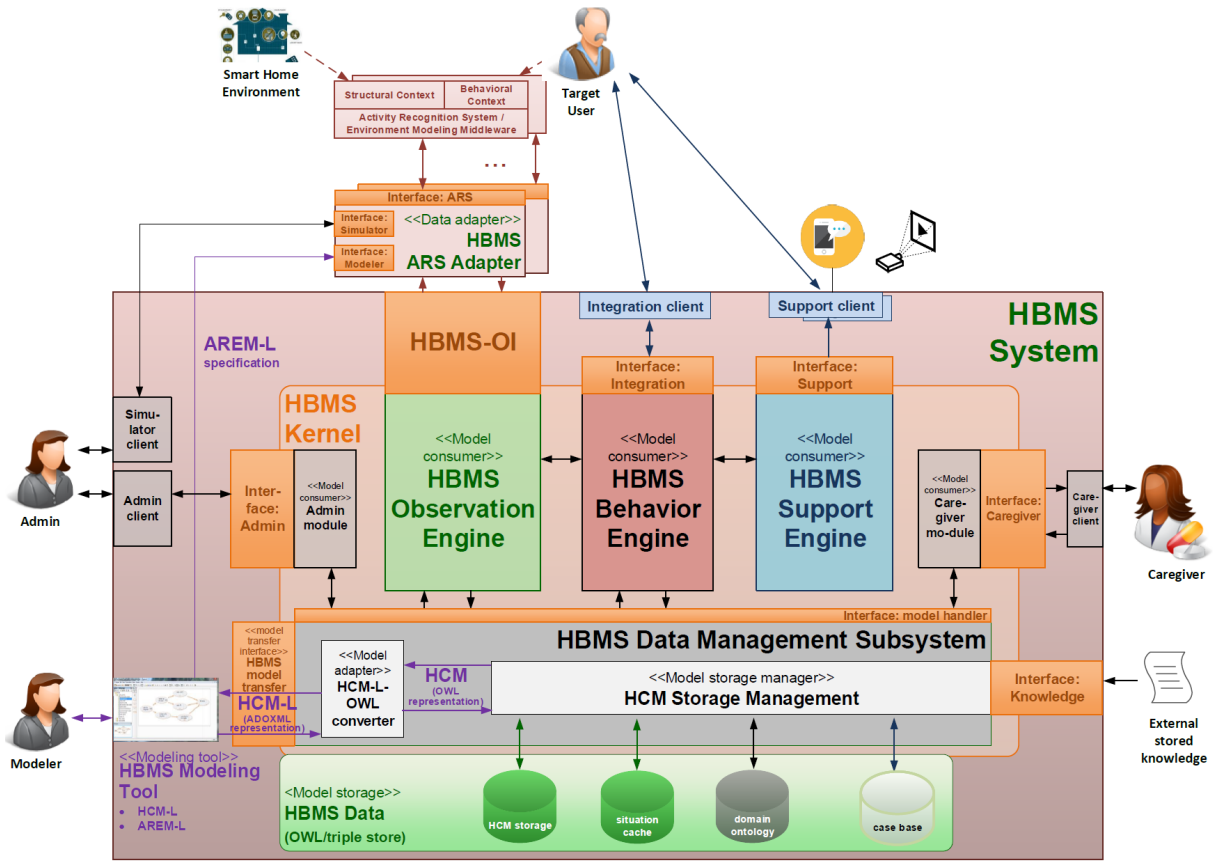



*Figure 10: Architecture 2017 (adapted from (Mayr et al. 2017))*

observation components (section 6.2), the HBMS-System intelligence (section 6.3) and the user interfaces (section 6.4) in more detail.

## 6.1 HCM-L Modeller

The HCM-L Modeller was one of the first system components realized, as we needed concrete models, created accordingly to the HCM-L meta-model definition, to be able to move on with the work on other components and to define interfaces.

A first prototype was realized with the Eclipse Rich Client Platform (RCP). The HCM meta-model was created in Eclipse Modelling Framework (EMF) format. EMF also supported "models to text" and "models to models" transformation (Oldevik et al. 2005), which was necessary for model validation and analysis purposes. For persistence layer implementation, we used a MySQL database and Hibernate.

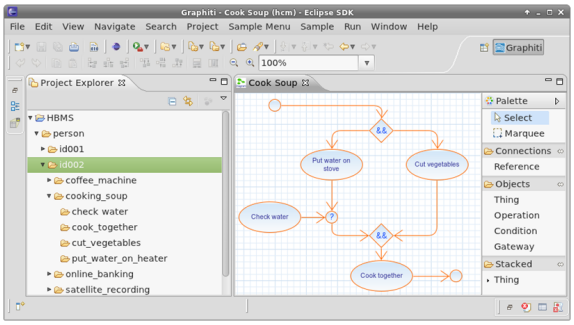



*Figure 11: Modelling Tool in Eclipse*

Figure 11 shows a screenshot of this prototype. As described in section 5, we had several severe changes in the meta-model of our DSML and realized, that such an incremental development process was connected with enormous workloads for changing the meta-model and graphical rep-

resentations in the Eclipse prototype. Moreover, some improvements to achieve a better end-user understanding were not feasible within its' graphical interface.

After this first prototypical requirements analysis, we performed a systematic collection of requirements for the next version of our HCM-L Modeller along the characteristics listed in (Kaschek and Mayr 1996).

Based here-on we decided to test the ADOxx Meta-Modelling Platform (Karagiannis and Kühn 2002) and finally chose it to implement the next version of HCM-L Modelling Tool (Michael et al. 2014, 2015).

The crucial factors for this decision were:

- The possibility of fast prototyping: a first stable version was completed in a month,
- Ease of changes in the meta-model and automatic tool adaptation,
- Availability of a simulation component,
- Easy to combine with external software, e.g. for reasoning (Al Machot et al. 2014),
- Availability of analysis functionalities,
- Easy to realize consistency checks,
- Highly professional general software and developer support.

The HCM-L meta-model elements were mapped to classes of the ADOxx meta-model and described using the proprietary ADOxx Library Language (ALL). ALL used the constructs defined in the ADOxx meta-meta-model (level M3) (Fill and Karagiannis 2013). ADOxx already included export and import functionalities in and from the ADOxx Description Language (ADL) format or a generic XML format. We adopted these to allow the transformation of HCM-L models to other representation formats, as used e. g. by inference or reasoning tools.

The HBMS team joined the *Open Modeling Initiative (OMI)* (see Karagiannis et al. 2007) and the related *Open Models Laboratory (OMiLAB)*: The HCM-L Modeller (see Figure 12) is, in a basic version, freely available at the OMiLAB webpage[1]. This version includes the manual creation of sequences, task context, structural context and BU models (BUMs) as well as macros. Furthermore, it is possible to step through BUMs and related sub-processes by using the graph analysis functionality of ADOxx. Model transfer between HCM-L Modeller and HBMS-System kernel was realized in this version via the adapted export functionalities mentioned above.

In 2016/17, the HCM-L Modeller was enhanced based on the MCA paradigm as a concretisation of the *Modelling Tool* architectural pattern (see Figure 8). As such concretisation, besides supporting the latest HCM-L constructs, it was able now to communicate with the HBMS kernel by transferring the XML representation of the selected subset of models (describing the context of the supported person) through the communication channel controlled by that kernel instance (e. g. represented as a web service call).

To control such transfers, we extended HCM-L with the new Configuration Workspace Model which connected the references to HBMS deployment sites (kernel installations) to the references to context models to be deployed at these sites. On modeller's demand, a model is employed to find the server over the network and conduct the model transfer by means of the deployment script.

## 6.2 Observation

The HBMS-System should be able to learn user's behaviour and to apply this knowledge to support the user unobtrusively in real-time. Thus, we investigated possibilities to observe users to recognizing their activities. Additionally, we developed an activity simulator tool which allowed us to generate sensor data for our scenarios via a web interface to be able to test the system independently from activity recognition systems.

### 6.2.1 Observation Interface

During the first project phase, we analysed existing user observation technologies and tools. The first HBMS-System prototype, however, was not able to automatically observe a user and to

[1] http://austria.omilab.org/psm/content/hcml/info

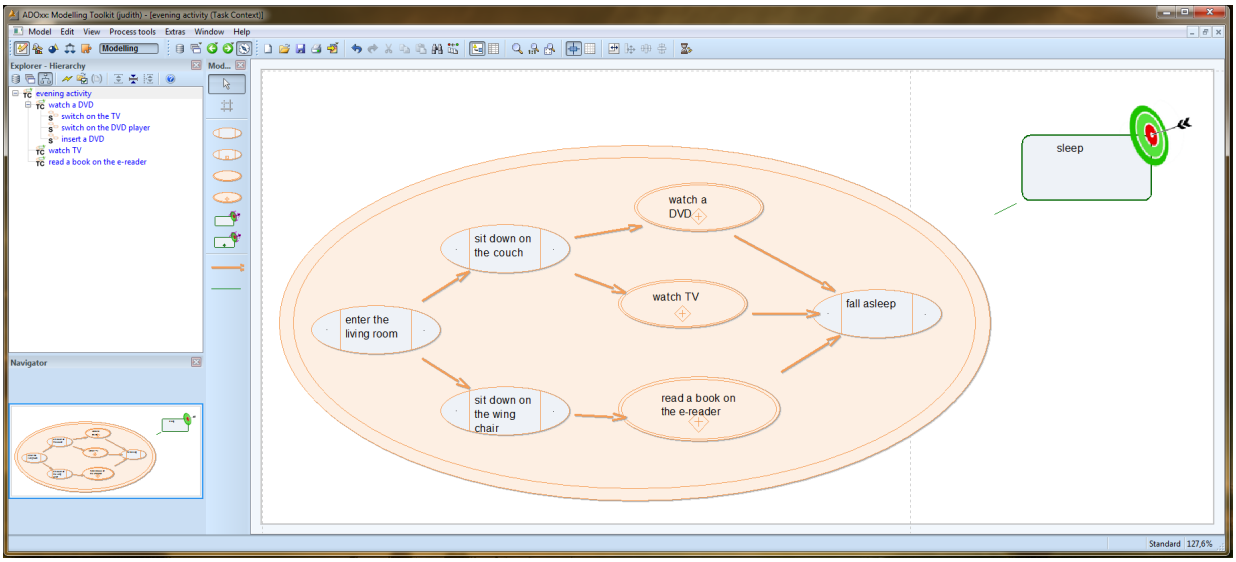



*Figure 12: HCM-L Modeller 2016*

detect his activities. The user behaviour had to be observed manually and communicated to the HBMS-System.

The first form of automatic observation was introduced in the HBMS-System prototype which was evaluated during LNF14. Simple wired sensors were connected directly to the HBMS-System, which also implemented a simple context management system component. This component was able to identify simple activities of a user and to forward them to provide required user support.

In developing the next observation prototype, we followed the MCA paradigm. Monitoring the user and handling the captured user context was outsourced to specialized context management middleware. The communication between this middleware and the HBMS kernel took place by means of the *HBMS observation interface (HBMS-OI)*. The observations obtained through this interface were subsequently processed and analysed by the HBMS observation engine, as shown in Figure 10.

We tested this HBMS-System prototype in our lab environment (see Figure 13) and used Nimbits[2] and FHEM[3] as middleware systems. To monitor user behaviour, we used a network of wired and wireless sensors supporting different protocols. The sensor network consisted of motion, pressure and contact sensors. All sensor data was gathered by means of the Arduino Coordinator [4] . The coordinator was responsible for identifying relevant sensor data changes and for writing the data corresponding to these changes to the relevant Nimbits channels. Also, the coordinator was responsible for comparing sensor values with given threshold values, e. g. if pressure sensor value exceeded 700 mbar it sent value “ON” or “1” to the relevant channel.

In our setup, for the identification purposes, we categorized Nimbits channels into two categories as toggling and firing channels.

Toggling channels were responsible for storing binary values, i.e. “1” or “0”, based on the ON/OFF value produced by the binary sensors. Toggling channels were mainly used to track events such as opening/closing doors, putting/lifting things etc. which typically generate binary values from the corresponding sensor devices. When the toggling channel detected a value change, e.g. from “0” to “1” or from “1” to “0”, it generated an event. For example, if a person “opens a door”, the

[2] https://www.nimbits.com/
[3] https://fhem.de/
[4] https://www.arduino.cc/

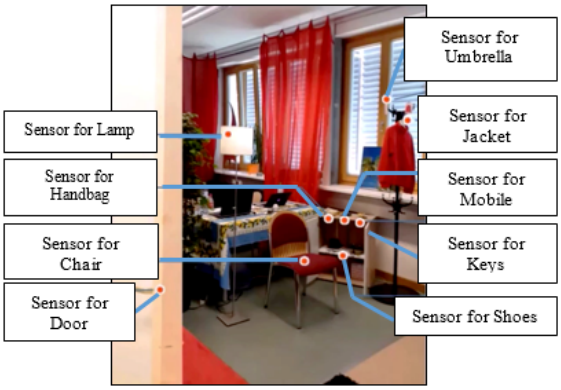



*Figure 13: Sensors in the Lab*

sensor connected to the door sent the corresponding signal to the coordinator, which stored the sent value in the given toggling channel. Consequently, the toggling channel generated an "open door" event and created meaningful semantic data to describe the event, combining it with the predefined meta-data.

Firing channels are used to store sensor data values which are typically static. Such channels use the channel timestamp change to generate a corresponding event. For example, a remote control button can be connected to the sensor which sends a predefined value on every click e.g. “user pressed the STOP button”. It is hard to recognize an activity by looking only at such value, so firing channels looks at the timestamp change to generate an event.

The problem with the approach described above was, that the low-level data available in a channel was directly accessed through the HBMS-OI. As a result, at that point in time the coupling between the middleware system and the HBMS-OI was still very high. Besides that, we were not satisfied with this solution, as only very simple atomic activities could be observed. Accordingly, we were looking for a way to make arbitrary activity recognition systems interoperable with HBMS-OI.

To overcome these deficiencies, we introduced a language to describe human behaviour observations called *AREM-L (Activity Recognition Environment Modelling Language)* (Mayr et al. 2017).

The main process of integrating the observations with HBMS-OI based on AREM-L is shown in Figure 14.

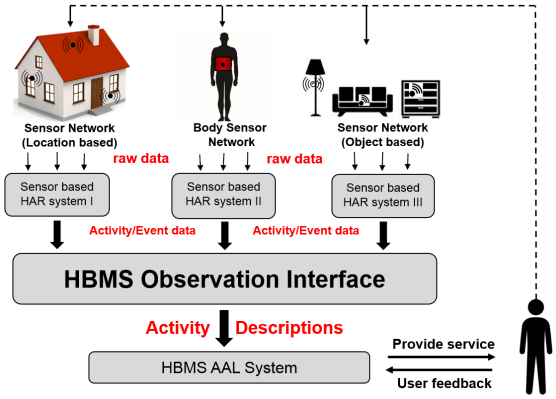



*Figure 14: Main processes of the HBMS-OI*

According to the MCA paradigm, in the current version of HBMS-OI, AREM-L is used to serve as a conceptual foundation for converting the data produced by the activity recognition systems (such as sensor based human activity recognition, HAR systems of different kind, as shown on Figure 14) to the semantic structures suitable for further processing in HBMS kernel (Mayr et al. 2017). The process includes the following steps:

1. Both human observation data and the contextual information are sent in real time from the HAR systems to the HBMS-OI;
2. HBMS-OI produces human observation recognition semantic structures out of this data based on the available AREM-L models;
3. These semantic structures are sent to the HBMS-System to be processed with a goal of providing the required assistance.

Evaluation activities to test the adaptivity and flexibility of the HBMS-OI are currently under way. Additionally, for the next future, we are going to evaluate the observation interface with video based activity recognition systems.

We are currently working on the integration of this version of the HBMS OI and the HBMS-System.

### 6.2.2 Simulation

Despite of all the successes with the observation interface for HBMS, we had to state the following: we had high expectations into current activity recognition capabilities in the beginning of HBMS but we had to realize at the end that current activity recognition approaches are too coarse to monitor all needed details for HBMS.

In order to overcome the weakness of the current activity recognition possibilities, we developed an activity simulator tool which allowed us to generate sensor data for our scenarios via a web interface. This generated data was forwarded to our context management middleware system in the same way as 'real' sensor data. This also allowed us to set up 'virtual labs' to test the HBMS-System. The HBMS simulator contained a set of simulator controls organized into control groups.

We distinguished the following types of simulator controls:

1. Toggling controls used to send one of the two alternative values when clicked, the value sent is the opposite of the current value.
2. Firing controls used to send the same value on every click.
3. Value-based controls used to send the user-selected value on request.

Figure 15 shows the controls which can send a value selected by means of radio button or select list. For every control, it was also possible to see the value which corresponded to the current value of a real sensor (if one was connected). This value was updated automatically when the sensor value was updated.

This tool allowed us also to simulate temperature and humidity meters to provide the flow of indoor and outdoor temperature and humidity data. The simulator provided the data as coming from these meters and was used to simulate different environmental settings, e.g. hot/cold or rainy/dry weather. The data from the simulator then went to our context management middleware which made it available to the HBMS-OI through additional specific channels (providing the currently simulated value when asked).

## 6.3 Intelligence

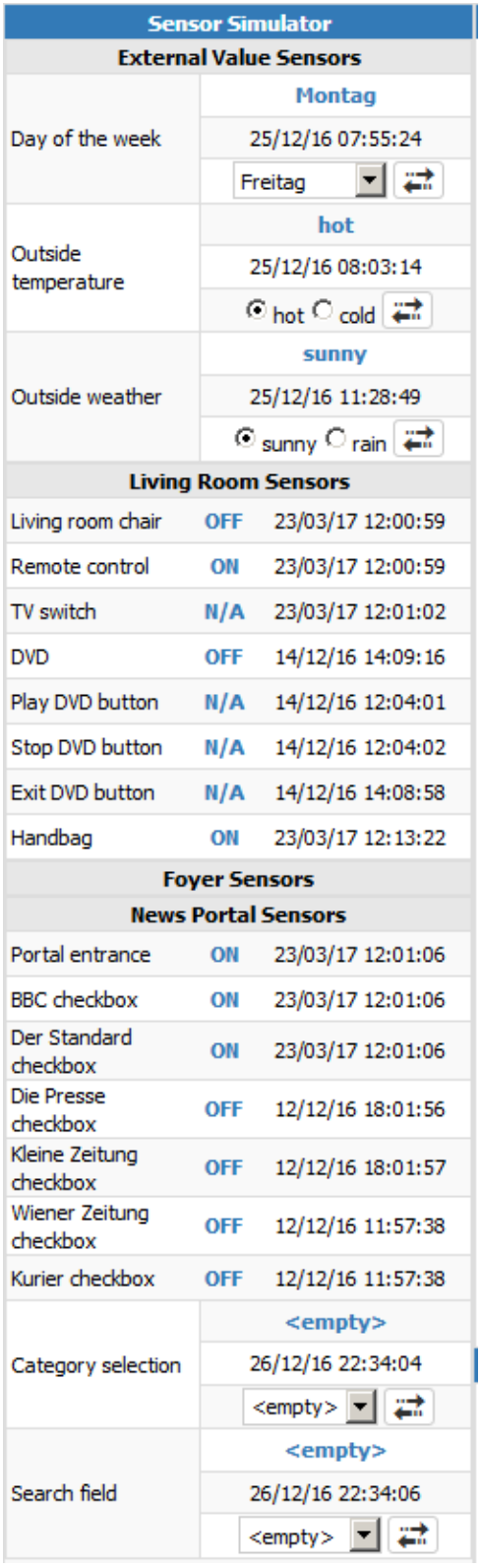
The first version of the HBMS-System had a *low intelligence level*: Models were created by hand including the integration of different models, only a part of the model concepts were translated into OWL, the structure accepted by the dialogue component was very tight and it included only text and no pictures or videos.

The next prototype of the HBMS-System included *basic reasoning mechanisms* which were included in the HCM-L Modeller. Sensors were connected with the HCM-L Modeller and the first support hints were presented at LNF14. Additionally, models were exported in XML format, which was used as an input for the knowledge base of the answer set programming solver. This reasoning module was able to calculate which next operation users should execute (see 'Reasoning for human support' in section 6.3.1 for details). Moreover, reasoning approaches for supporting activity recognition were tested on laboratory datasets (see 'Reasoning for Human Activity Recognition' in section 6.3.1 for details).

The *latest prototype*, presented at LNF16, included the matching of detected actions with the actions in the knowledge base and a direct visualization of the results. The matching component tries to find observed behaviour in formerly detected and saved behavioural units and returns if the action was correct, wrong in relation to its order in a behavioural unit or the conditions performed, a reverse action or a yet unrecognised one (see section 6.3.2 for more details). The Visualiser, as part of the kernel monitoring client, provides a view on the current state of the matching session (see section 6.3.3 for more details).

### 6.3.1 Reasoning

The reasoning within the HCM-L Modeller has been divided into two major modules; (a) the reasoning module to recognize human activity and (b) the reasoning to choose the optimal step



| Sensor Simulator | | |
|---|---|---|
| **External Value Sensors** | | |
| Day of the week | Montag | |
| | 25/12/16 07:55:24 | |
| | Freitag | |
| Outside temperature | hot | |
| | 25/12/16 08:03:14 | |
| | hot / cold | |
| Outside weather | sunny | |
| | 25/12/16 11:28:49 | |
| | sunny / rain | |
| **Living Room Sensors** | | |
| Living room chair | OFF | 23/03/17 12:00:59 |
| Remote control | ON | 23/03/17 12:00:59 |
| TV switch | N/A | 23/03/17 12:01:02 |
| DVD | OFF | 14/12/16 14:09:16 |
| Play DVD button | N/A | 14/12/16 12:04:01 |
| Stop DVD button | N/A | 14/12/16 12:04:02 |
| Exit DVD button | N/A | 14/12/16 14:08:58 |
| Handbag | ON | 23/03/17 12:13:22 |
| **Foyer Sensors** | | |
| **News Portal Sensors** | | |
| Portal entrance | ON | 23/03/17 12:01:06 |
| BBC checkbox | ON | 23/03/17 12:01:06 |
| Der Standard checkbox | ON | 23/03/17 12:01:06 |
| Die Presse checkbox | OFF | 12/12/16 18:01:56 |
| Kleine Zeitung checkbox | OFF | 12/12/16 18:01:57 |
| Wiener Zeitung checkbox | OFF | 12/12/16 11:57:38 |
| Kurier checkbox | OFF | 12/12/16 11:57:38 |
| Category selection | <empty> | |
| | 26/12/16 22:34:04 | |
| | <empty> | |
| Search field | <empty> | |
| | 26/12/16 22:34:06 | |
| | <empty> | |

*Figure 15: Simulator including different 'sensors'*

for human support based on HCM-L models. The aim of the first module is to identify the activity of the observed person to advise her/him about what is the optimal operation to be executed using the second module.

**Reasoning for Human Activity Recognition.** Regarding human activity recognition, different approaches have been developed in the frame of the HBMS project independently to be applied for real life scenarios. The proposed approaches show promising results using not only the HBMS laboratory dataset but also the well-known human activity CASAS dataset (Cook et al. 2013). The datasets are collected based on real life scenarios using different types of non-intrusive sensors that are mounted in real smart homes. The approaches can be summarized as follows:

1. A reasoning method based on answer set programming that uses different types of features for selecting the optimal sensor set, and a fusion approach to combine the beliefs of the selected sensors using an advanced evidence combination rule of Dempster–Shafer theory (Al Machot et al. 2018a) (see Figure 16).
2. A reasoning method based on a windowing approach for analysing sensor data to identify the best fitting sensors that should be considered in the selected window of sensor events. The second contribution proposes a set of different statistical spatio-temporal features to recognize human activities using Support Vector Machines (Al Machot et al. 2017).
3. A human activity recognition approach based on a Recurrent Neural Networks (RNN) model, which is trained based on dynamic systems perspective during the weight initialization process (Al Machot et al. 2018b).

**Reasoning for human support.** Regarding the human support, an approach based on Answer Set Programming (ASP) has been proposed. The aim was to solve an optimization problem depending on the following factors; (a) the importance of performing an operation according to the user history; (b) the cost value of choosing an operation

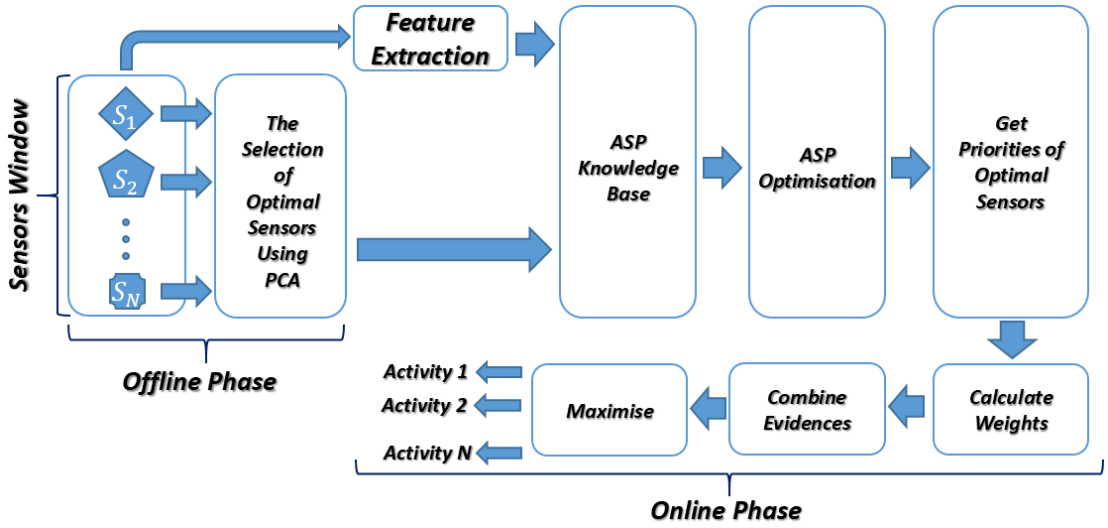



*Figure 16: Reasoning system structure for activity recognition*

based on the similarity between the current user profile and other users; (c) the time when the operation should be performed (Al Machot et al. 2014).

### 6.3.2 Matching

Another important development step was the implementation of the matching component *(HBMS matcher)*. Its aim is to match the semantic structures corresponding to the recognitions of the observed behaviour against the current HCM, to find a behavioural scenario (behavioural unit) including the observed action and a position of the action inside this scenario. Finding such match is a prerequisite for predicting next actions and providing support.

The matcher recognizes the following four categories of actions based on the recognition information coming from the observation engine:

1. Correct actions which are performed correctly based on the given BU;
2. Wrong actions which are present in the model but have been performed in the wrong place or under wrong conditions (i. e.. taking the shoes before going to the foyer);
3. Reverse actions which has been performed to revert the effect of the previous (possibly incorrect, but also correct) action e. g. "put the handbag back ";
4. Unrecognised actions which are not present in the model.

We based the matching algorithm on two main categories of checks:

1. Finding the set of possible matches based on changes in state (*state-based checks*). This is supported through accompanying the operations in the model with capturing state snapshots: every such snapshot describes the set of state configurations which are considered correct for entering the specific operation (capturing pre-operation snapshot) and for exiting this operation (capturing post-operation snapshot). The match is found by comparing the observed state snapshots of the observation data structure coming from the observation engine with the capturing snapshots of all operations defined for the available behaviour models.
2. Narrowing or modifying the set of matches based on checking preconditions (*condition-based checks*). This is supported through accompanying the operations in the model with parsed precondition specifications which include the internal representation of the pre-

condition code. The conditions from the possible matched operation are checked for the observation data structure, the match is found if this condition evaluates to true as a result of this check.

Based on the above set of checks, the correct and wrong actions can be recognized as follows:

1. The action is recognized as correct if the state coming with the observation completely matches the state accompanying the particular snapshot in the model, the changes introduced into the state are the same for both observation and the model operation, and the preconditions for this action are fulfilled.
2. The incorrect actions are recognized by matching the changes introduced into the state but missing the match for the precondition and post-condition state as a whole.

All recognized actions are encapsulated into the match result structures containing the type (correct, wrong etc.) and the timestamp of the match, and with the information about the matched operation and the existing reverse operation; the matcher also interacts with the prediction component to include the information about the predicted next operation into the match result.

As a part of the matching session, the matcher maintains two separate session queues collecting *"match result"* objects: the right action queue for correct actions; and the wrong action queue for the incorrect actions. The reverse actions are not added to these queues, they lead to deleting the last action from the queue. As a result, it is possible for the user to revert the damage caused by incorrect actions by performing the reverse actions in the reverse order, and go back as much steps as he/she wishes.

The match result is transferred to the support engine to serve as a basis for providing support.

### 6.3.3 Visualising

The next step was implementing the *kernel monitoring client*. This client is shared by both admin and caregiver interfaces of the kernel. It allows to monitor the events processed by the kernel, execute simulations of the sensor events, and perform administrative activities. It is implemented as an AngularJS client application communicating with the kernel by means of JSON-based API.

The main part of the monitoring client is the *HBMS Visualizer*. It allows for the user to see the current state of the matching session by:

1. Showing the graphical representation of the available BUs;
2. Reacting to the changes initiated by the HAR system through the HBMS-OI by highlighting specific diagram elements;
3. Providing a scalable display making possible to zoom to a specific subset of operations.

The Visualizer supports six different states of the operation element and three states of the goal element, a state (e. g. “correctly/wrongly matched as a session’s past operation”, “correctly/wrongly matched as a current operation”, “predicted”) corresponds to the specific colour and thickness of the element’s outline.

The Visualizer queries the state of the matching session several times per second and changes the display based on the obtained information. As a result, it is possible to see:

1. The position of the user in the current BU;
2. If the user made a mistake in handling the scenario;
3. The set of operations the user has been performed in the current session;
4. The number of steps the user has to go back to fix the wrong actions;
5. The predicted next operation or operations;
6. If the user has reached the goal.

On Figure 17, the user has just performed "stand up from the chair" operation after correctly performing "take the sunny weather shoes" and other preceding operations in the behavioural unit, the "take the keys" operation is shown as predicted.

The Visualiser also allows switching between behavioural units by means of a list of graphical

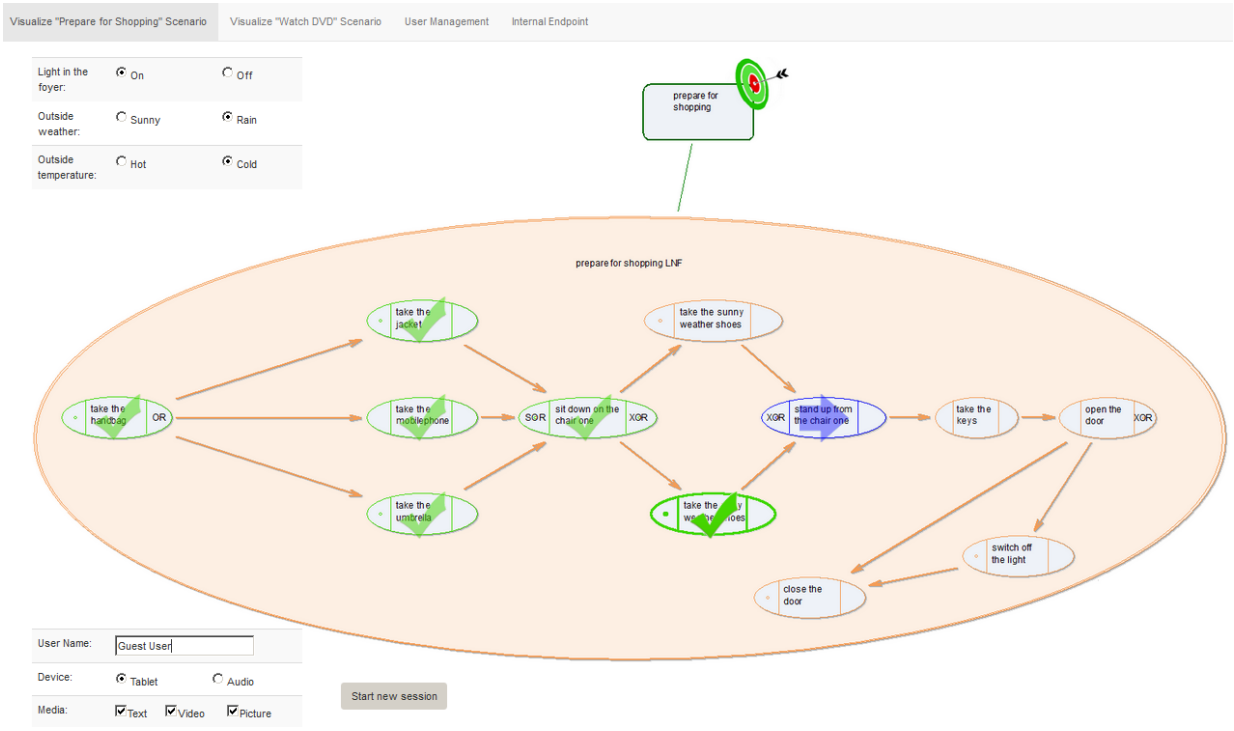



*Figure 17: HBMS Visualizer showing the current state of the matching session*

depictions of the behavioural unit models (at the bottom of Figure 17). Every such depiction is a scaled down version of the Visualizer display, it is updated dynamically when matching activities are performed by the kernel.

## 6.4 User Interface

The first dialogue component of HBMS to help users interactively and unobtrusive in performing their activities was based on first results of our requirements analysis. This HBMS *'Support 1'* was developed as a prototype for user evaluations in the context of a master thesis. It was supported by android tablets and mobiles and developed using PhoneGap, OSGi and webservice technologies.

The texts and next steps were derived in a lean OWL representation directly from the conceptual models in the HCM. This information was transformed into a representation of one single step and navigation possibilities to possible following steps (see Figure 18). Existing text-to-speech technologies were used to provide a voice output. This first support client prototype was used to evaluate the HBMS-System in LNF12: The participants used a coffee machine in our station at one end of the university campus, where their behaviour was manually modelled by team members. Afterwards, they moved to our station at the other end of the university campus where exactly these sequences were presented step-by-step on a tablet or a mobile phone.

Learning from the first user feedbacks (e. g. variable text sizes, louder voice outputs, including graphics) we developed *enhanced support clients* for HBMS and their appropriate user interfaces. We have evaluated these ideas in 2013 by developing different user interface mock-ups, which were tested in a design study (Strobl and Katzian 2014) and stronger discussed in a master thesis. Figure 19 presents one of the screens for the evaluation.

For the LNF14 further improvements of the user interface were provided including the outcomes of the design study: To stay platform independent and interoperable, the second HBMS support prototype was realized as a *web application* using the


*Figure 18: First UI prototype*


*Figure 19: Mock-Ups of the UI for evaluations*

.NET framework and JQuery Mobile. The communication of this support client and the current HBMS core system was done via XML interfaces. Thus, this web-based solution was presentable *on every device with an internet browser*. Media data was already included and there was an acoustic support possibility provided, which started automatically. Nevertheless, the speech output was presented on another device to be able to synchronize it with the texts.

As the HBMS architecture developed over time to a *Model Centred Architecture*, we enhanced our mobile support clients and made them additionally available as *android native apps*. In doing so, mobile functions for audio, vibration and localization enhanced the support quality: it is possible to start speech output automatically based on default setting, vibration feedback is possible, it is possible to zoom into explanations and pictures and it is easier to adopt the graphics accordingly on each android device. In an additional navigation bar, preferences for support could be selected and personal context data is presented in a user friendly way. The prototype was again tested at LNF16.

Moreover, we investigated devices like robots, smart watches and smart TV-devices to be used for user support connected to the HBMS-System via a support client interface and made a proof-of-concept for a support scenario with Amazon Echo as an output device including the language assistant Alexa.

## 7 Ideas and Future prospects

The HBMS project team can look back on nearly 7 successful years. What started with a *visionary idea* resulted in a *stable beta-version of an active assistance system*.

**The impact.** Within HBMS, 1 PhD thesis and 6 master theses were completed. 4 PhD theses and 4 master theses are under development. 12 school students were involved in HBMS during their holiday work placement. The HBMS-System and its components and ideas were evaluated with more than 250 people in 4 prototype evaluations, 2 workshops with 60 people, 2 online surveys with more than 540 people, a user design study with 55 people and 2 qualitative studies with more than 100 participants. Since 2014, our modelling approach was taught to more than 200 students at the Next Generation Enterprise Modelling (NEMO) summer school. The project members published more than 25 papers at conferences and articles in journals, held tutorials on Ambient Assistance

and Modeling as well as on IT- Based Ambient Assistance and initialized the AHA workshop series (ER-Workshop on Conceptual Modeling for Ambient Assistance and Healthy Ageing).

Nevertheless, the work on the project vision will continue in future:

**DSMM and the HCM-L Modeller.** Further development of the modelling method is always connected with enhancement of the modelling tool. Aspects, which are important for the DSMM are further investigations of the concept goal (see e. g. goal modelling languages in requirements engineering as iStar (Yu 1997), GoalML (Overbeek et al. 2015) or (Rolland and Salinesi 2005)), the grounding of the HCM-L in a foundational ontology (see e. g. Guizzardi et al. 2008) or taking ideas of multi-perspective modelling (Frank et al. 2017) into consideration. The graphical notation has to be adapted accordingly to additions in the meta-model, e. g. for already done refinements of the context (Michael and Steinberger 2017). Additionally, the HCM-L modeller has to be adapted accordingly to the changes of the meta-model. A new version of the modelling tool should be provided at the OMiLAB website. Other open issues are related with the graphical interface of the modelling tool: a 3D visualisation for BUs and related structural elements would support end users to better understand their preserved behaviour.

**Observation.** For the next future, further evaluations are planned, to test the observation interface with video based activity recognition systems.

**Intelligence.** The semantic annotation of web-user interfaces would make it possible to provide meaningful support of their use. Furthermore, the development of the simulator component would make an the automated creation of the simulation interface by using social context information possible. Another missing aspect is the integration of sequences of behaviour models into existing ones in the knowledge base. To finalize this aspect, process mining approaches as e. g., (van der Aalst 2016) might be helpful.

**User Interfaces.** Further improvements onto the development of active multimodal support for human behaviour include a DSML for multimodal support, the test of further devices e. g., augmented reality approaches, and evaluations of these devices together with end users.

**Analysis and Evaluation.** An ongoing state-of-the-art analysis and regular evaluations of the technical prototypes are taken for granted. Currently, we are planing the evaluation of various notations for modelling the advanced HBMS context, namely user models, environment models and spatial models.

Moreover, the developed HBMS-System is applicable in several other domains, e. g. *medicine* (Czaplik et al. 2016), *cyber-physical systems* (Berardinelli et al. 2017), *production systems in the area of Industry 4.0* or *IoT applications*. Further investigations in these areas are planned.

Last but not least, such a system is not developed with so much ambition that it remains only in the prototype stage and (as so often in scientific developments) is forgotten after a few years. We hope to be able to bring it to market in the next few years together with a development partner, so that the HBMS system can fulfil its original vision and be *helpful for people*.

## Acknowledgments

*We would like to thank Heinrich C. Mayr for his visionary ideas, his engagement in raising project funds, his critical feedback and for being our project and team leader as well as scientific mentor.*

## References

Al Machot F., Haj Mosa A., Ali M., Kyamakya K. (2017) Activity Recognition in Sensor Data Streams for Active and Assisted Living Environments. In: IEEE Transactions on Circuits and Systems for Video Technology

Al Machot F., Mayr H. C., Michael J. (2014) Behavior Modeling and Reasoning for Ambient Support: HCM-L Modeler. In: Proceedings of the International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA-AIE 2014). Lecture Notes in Artificial Intelligence

Al Machot F., Mayr H. C., Ranasinghe S. (2018a) A Hybrid Reasoning Approach for Activity Recognition Based on Answer Set Programming and Dempster-Shafer Theory. In: Recent Advances in Nonlinear Dynamics and Synchronization. Springer, pp. 303–318

Al Machot F., Ranasinghe S., Plattner J., Jnoub N. (2018b) Human Activity Recognition based on Real Life Scenarios. In: Proc. of the IEEE International Conference on Pervasive Computing and Communications, CoMoRea workshop. IEEE

Berardinelli L., Mazak A., Alt O., Wimmer M., Kappel G. (2017) Model-Driven Systems Engineering: Principles and Application in the CPPS Domain In: Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects Biffl S., Lüder A., Gerhard D. (eds.) Springer International Publishing, pp. 261–299 https://doi.org/10.1007/978-3-319-56345-9_11

Bonfiglio S., Bekiaris E., Panou M., Sala P., Bautista J., Agües M., Robledo M. G., Van Isacker K., Cabrera F., Mixco V. J., de la Maza C., Staehli B. (2008) Use Cases and application scenarios for independent living applications

Cook D. J., Crandall A. S., Thomas B. L., Krishnan N. C. (2013) CASAS: A smart home in a box. In: Computer 46 (7)

Czaplik M., Nazari P. M. S., Roth A., Rumpe B., Voigt V., von Wenckstern M., Wortmann A. (2016) Der Weg zur Modellbasierten Evolution und Adaption medizinischer Leitlinien. In: Software Engineering Workshops 2016. CEUR-ws.org, pp. 195–200

Fill H.-G., Karagiannis D. (2013) On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform. In: Enterprise Modelling and Information Systems Architectures Vol. 8, pp. 4–25 http://eprints.cs.univie.ac.at/3657/1/Fill_Karagiannis_EMISA_2013.pdf

Fliedl G., Gratzer W., Strobl T., Winkler C. (2013) Mobile Instruction Apps Based on Linguistic Text Reduction. In: Rault J.-C. (ed.) Proceedings of ICSSEA 2013

Frank U. (2013) Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines. In: Reinhartz-Berger I., Sturm A., Clark T., Cohen S., Bettin J. (eds.) Domain Engineering. Springer Berlin Heidelberg, Berlin and Heidelberg, pp. 133–157

Frank U., Reinhartz-Berger I., Sturm A., Clark T. (2017) A Multi-level Approach for Supporting Configurations: A New Perspective on Software Product Line Engineering. In: Cabanillas C., España S., Farshidi S. (eds.) Proc. of the ER Forum 2017 and the ER 2017 Demo Track co-located with the 36th International Conference on Conceptual Modelling (ER 2017), pp. 170–178

Guha R., Brickley D., Macbeth S. (2016) Schema.org: Evolution of structured data on the web. In: Communications of the ACM 59 (2), pp. 44–51

Guizzardi G., Falbo R., Guizzardi R. S. (2008) Grounding Software Domain Ontologies in the Unified Foundational Ontology (UFO): The case of the ODE Software Process Ontology. In: Lencastre M., Falcão e Cunha, João: Valecillo, Antonio (eds.) 11th Iberoamerican Workshop on Requirements Engineering and Software Environments, pp. 127–140

Karagiannis D., Grossmann W., Höfferer P. (2007) Open Model Initiative: A Feasibility Study. www.openmodels.at

Karagiannis D., Kühn H. (2002) Metamodelling Platforms. In: Bauknecht K., Tjoa A. M., Quirchmayr G. (eds.) E-Commerce and Web Technologies. LNCS Vol. 2455. Springer, p. 182

Kaschek R., Mayr H. C. (1996) A characterization of OOA tools. In: IEEE International Symposium on Assessment of Software Tools, pp. 59–67

Katz S. (1983) Assessing self-maintenance: Activities of daily living, mobility, and instrumental activities of daily living. American Geriatrics Society, New York NY

Kofod-Petersen A., Cassens J. (2006) Using Activity Theory to Model Context Awareness. In: Roth-Berghofer T., Schulz S., Leake D. (eds.) Modeling and Retrieval of Context. Lecture Notes in Computer Science Vol. 3946. Springer, pp. 1–17 http://dx.doi.org/10.1007/11740674_1

Kop C., Mayr H. C. (1998) Conceptual predesign bridging the gap between requirements and conceptual design. In: 1998. Proceedings. 1998 Third International Conference on Requirements Engineering, pp. 90–98 http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=667813

Lawton M. P., Brody E. M. (1969) Assessment of older people: self-maintaining and instrumental activities of daily living. In: The Gerontologist 9(3), pp. 179–186

Leont'ev A. N. (1978) Activity, Consciousness, and Personality. Prentice-Hall, Englewood Cliffs, NJ

Lindland O., Sindre G., Solvberg A. (1994) Understanding quality in conceptual modeling. In: IEEE Software 11(2), pp. 42–49

Mayr H. C., Al Machot F., Michael J., Morak G., Ranasinghe S., Shekhovtsov V., Steinberger C. (2016) HCM-L: Domain-Specific Modeling for Active and Assisted Living. In: Karagiannis D., Mayr H. C., Mylopoulos J. (eds.) Domain-specific conceptual modeling. Springer, pp. 527–552 http://link.springer.com/chapter/10.1007/978-3-319-39417-6_24

Mayr H. C., Michael J. (2012) Control pattern based analysis of HCM-L, a language for cognitive modeling. In: International Conference on Advances in ICT for Emerging Regions (ICTer2012). IEEE, pp. 169–175

Mayr H. C., Michael J., Ranasinghe S., Shekhovtsov V. A., Steinberger C. (2017) Model Centered Architecture In: Conceptual Modeling Perspectives Cabot J., Gómez C., Pastor O., Sancho M. R., Teniente E. (eds.) Springer International Publishing, pp. 85–104 https://doi.org/10.1007/978-3-319-67271-7_7

Mayring P. (2010) Qualitative Inhaltsanalyse. In: Mey G., Mruck K. (eds.) Handbuch Qualitative Forschung in der Psychologie. VS Verlag für Sozialwissenschaften, Wiesbaden, pp. 601–613

Michael J., Al Machot F., Mayr H. C. (2014) A Behavior Centered Modeling Tool Based on ADOxx. In: Proceedings of the CAiSE'14 Forum. CEUR

Michael J., Al Machot F., Mayr H. C. (2015) ADOxx based Tool Support for a Behavior Centered Modeling Approach. In: Proceedings of the 8th PErvasive Technologies Related to Assistive Environments (PETRA) conference. ACM

Michael J., Grießer A., Strobl T., Mayr H. C. (2013) Cognitive Modeling and Support for Ambient Assistance. In: Mayr H. C., Kop C., Liddle S., Ginige A. (eds.) Information Systems: Methods, Models, and Applications: Revised Selected Papers.. LNBIP Vol. 137. Springer, Heidelberg

Michael J., Mayr H. C. (2013) Conceptual Modeling for Ambient Assistance. In: Ng W., Storey V. C., Trujillo J. (eds.) Conceptual Modeling - ER 2013. Lecture Notes in Computer Science (LNCS) Vol. 8217. Springer, pp. 403–413

Michael J., Mayr H. C. (2015) Creating a Domain Specific Modelling Method for Ambient Assistance. In: International Conference on Advances in ICT for Emerging Regions (ICTer2015). IEEE, pp. 119–124

Michael J., Mayr H. C. (2017) Intuitive understanding of a modeling language. In: Proceedings of the Australasian Computer Science Week Multiconference (ACSW'17). ACM, New York NY, USA, pp. 1–10

Michael J., Steinberger C. (2017) Context Modeling for Active Assistance. In: Cabanillas C., España S., Farshidi S. (eds.) Proc. of the ER Forum 2017 and the ER 2017 Demo Track co-located with the 36th International Conference on Conceptual Modelling (ER 2017), pp. 221–234

Moody D. (2009) The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. In: IEEE Transactions on Software Engineering 35(6), pp. 756–779

Object Management Group OMG: (n.d.) Meta Object Facility (MOF) Core. last accessed 2018-01-12 http://www.omg.org/spec/MOF/

Oldevik J., Neple T., Grønmo R., Aagedal J., Berre A.-J. (2005) Toward Standardised Model to Text Transformations. In: Hartman A., Kreische D. (eds.) Model Driven Architecture – Foundations and Applications. LNCS Vol. 3748. Springer Berlin, Heidelberg, pp. 239–253

Overbeek S., Frank U., Köhling C. (2015) A language for multi-perspective goal modelling: Challenges, requirements and solutions. In: Computer Standards & Interfaces 38, pp. 1–16

Ranasinghe S., Al Machot F., Mayr H. C. (2016) A review on applications of activity recognition systems with regard to performance and evaluation. In: International Journal of Distributed Sensor Networks 12(8)

Rolland C., Salinesi C. (2005) Modeling Goals and Reasoning with Them. In: Aurum A., Wohlin C. (eds.) Engineering and Managing Software Requirements. Springer-Verlag, pp. 189–217

Schütte R. (1998) Vergleich alternativer Ansätze zur Bewertung der Informationsmodellqualität. In: Fachtagung: Modellierung betrieblicher Informationssysteme, Fachgruppe MobIS

Shekhovtsov V. A., Mayr H. C., Ranasinghe S. (2018) Model-Based Interfacing to Activity Recognition

Steinberger C., Michael J. (2018) Towards Cognitive Assisted Living 3.0. In: Advanced Technologies for Smarter Assisted Living solutions: Towards an open Smart Home infrastructure (SmarterAAL workshop) at Percom 2018. IEEE, pp. 1–6

Strobl T., Katzian A. (2014) Darstellungsvarianten für handlungs-unterstützende Apps: Ergebnisse einer Evaluationsstudie im Rahmen des Projekts HBMS. In: Wohnen-Pflege-Teilhabe – 7. Deutscher AAL-Kongress. VDE

Strube G. (1996) Wörterbuch der Kognitionswissenschaft. Klett-Cotta, Stuttgart

Tulving E. (1972) Episodic and semantic memory. In: Tulving E., Donaldson W., Bower G. H. (eds.) Organization of memory. Academic Press, New York, pp. 381–403

van der Aalst W., ter Hofstede A., Kiepuszewski B., Barros A. (2003) Workflow Patterns. In: Distributed and Parallel Databases 14(1), pp. 5–51

van der Aalst W. M. P. (2016) Process Mining: Data Science in Action, 2nd ed. Springer https://doi.org/10.1007/978-3-662-49851-4

Wöckl B., Yildizoglu U., Buber-Ennser I., Aparicio Diaz B., Tscheligi M. (2011) Elderly Personas: A Design Tool for AAL Projects focusing on Gender, Age and Regional Differences. In: Partnerships for Social Innovations in Europe

Wohed P., van der Aalst W. M., Dumas M., ter Hofstede A. H., Russell N. (2005) Pattern-based Analysis of BPMN. In: http://eprints.qut.edu.au/2977/

Yu E. S. K. (1997) Towards modelling and reasoning support for early-phase requirements engineering. In: Proceedings of the Third IEEE International Symposium on Requirements Engineering, pp. 226–235

# Enterprise Modelling and Information Systems Architectures

Enterprise Modelling and Information Systems Architectures – International Journal of Conceptual Modeling (EMISA) is a scholarly peer-reviewed open access journal with a unique focus on novel and innovative conceptual modeling research and its applications.

EMISA is the leading international scholarly journal for state-of-the-art conceptual modeling research. EMISA publishes thoughtful, well-developed articles on all facets of analysing, designing, investigating, evaluating and applying conceptual models, enterprise models, enterprise and information systems architectures, corresponding modeling languages and modeling methods, and is open to submissions from all scientific disciplines and research fields. The editorial board imposes no restrictions regarding the research paradigm or research method, encourages multi- and transdisciplinary research contributions, and welcomes submissions from for-profit, nonprofit and government organisations in a dedicated Industry/Experience Report section.

The journal will address (but will not limit itself to) the following specific areas:

- General Purpose and Domain-Specific Modeling Languages
- Modeling Methods, Metamodeling, Method Engineering
- Enterprise Architectures, Information Systems Architectures
- Ontologies and Reference Models
- Analysis Patterns, Design Patterns, Architectural Patterns
- Analysis of Conceptual Models and Modeling Languages
- Evaluation and Quality of Conceptual Models, Architectures, and Languages
- Model-Driven Development, Models@run-time, Executable Models
- Business Process Management
- Enterprise Architecture Management
- Information Systems Design and Design Methods
- Information Systems Development Methods and Tools
- Software Tools for Conceptual and Enterprise Modeling, Business Process Management and Enterprise Architecture Management
- Learning and Teaching Conceptual Modeling
- Applications of Conceptual Modeling in Organizations
- Innovative Approaches to Conceptual Modeling

EMISA is a publisher-independent journal published by the German Informatics Society (GI) and is a publication of its Special Interest Group on Modeling Business Information Systems (SIG MoBIS) and its Special Interest Group on Design Methods for Information Systems (SIG EMISA).

https://emisa-journal.org

ISSN 1866-3621 (Online)