Ulrich Frank

A Conception of a Presentation Development and Management System Featuring 'Smart Slides'

This paper presents a conception of a presentation development and management system (PDMS) and a related process model that guides its adequate use. It is inspired by the idea to replace certain graphical representations on presentation slides with diagrams that are constructed with domain-specific modelling languages (DSML) and corresponding model editors. The PDMS features an extensible set of graphical DSML that originate in a method for enterprise modelling. They provide the conceptual foundation for graphical representations that include domain-specific semantics. Specifying DSML and generating code for corresponding model editors is supported by an integrated metamodelling environment. Various DSML can be combined to create interactive, multi-language diagrams. The embedded semantics enables versatile machine analysis and allows for interactive slides. An accompanying compound architecture allows for integrating interactive diagrams with slides that contain traditional types of content such as text or drawings. Furthermore, it provides the conceptual foundation for storing presentations in a steadily growing common repository of organisational knowledge, thereby promoting reuse on various levels of abstraction.

1 Introduction

Presentation slides are an important medium to foster communication in many business scenarios. Often, especially in consulting firms, the professional design of presentation slides is regarded as an important prerequisite for illustrating problems and project results. As a consequence, presentation software is an important instrument for structuring and representing problem domains in business. Respective tools have matured over several years and allow for the creation of impressive presentations. While we do not know of a study on the scale and the economics of designing, using and maintaining business presentations, it seems reasonable to suppose that many firms have accumulated a large amount of slides. As a consequence, one can assume that the creation and maintenance of presentation slides causes remarkable costs – and that corresponding presentation tools are probably regarded as an instrument of outstanding relevance by many. However, despite the remarkable convenience and functionality offered by these tools as well as the impressive look of professionally developed business presentations, current practice is far from being satisfactory.

Occasionally, the use of presentation software – especially of PowerPoint© – has been criticised for corrupting creativity and individual presentation styles, thus contributing to boring presentations that compromise a 'contemplative analytical method' (Tufte 2006, p. 6). To counter this effect, Tufte emphasises the relevance of a 'cognitive approach' that should not be dominated by the 'limitations of the presentation technology' (ibid). While it is the question whether boring presentations should be contributed to the software or rather to the author, from an information systems perspective further aspects are more relevant anyway. They relate to productivity, quality, integrity and reuse.

On the one hand, current shortcomings are affected by the technology that is used to manage presentations as sequences of slides: Usually, presentations are stored as files. Whenever a new presentation is created, reuse is restricted to

copying slides from an existing file to another – or to copy a file and afterwards modify the duplicate. The resulting redundancy is a serious threat to integrity and may increase maintenance costs tremendously. On the other hand, productivity, quality and integrity suffer from a missing conceptual foundation of the content that is represented in slides. In other words: The tools that are used to produce slides and the content of the slides themselves lack formal semantics (in the remaining text, the use of the term 'semantics' will always relate to formal semantics). The text that is used within slides does not include much formal semantics. Nevertheless it allows for certain kinds of analysis such as retrieval and spell or grammar checking. The lack of semantics is a particular problem for graphical representations. If they are not just bitmaps, they are at best structured and weakly typed drawings. They are structured, if they are composed of various elements that can be manipulated separately in terms of presentation style or animation. They are typed because there are different types of presentation objects – such as text, graphical shapes, tables, video etc. – each of which is characterised by a certain set of operations. Typing is, however, restricted to presentation issues. It does not account for the semantics of the represented objects. The lack of semantics, i.e., the lack of rules that constraint the use of objects, promotes flexibility: Users can draw whatever they like. However, it also reveals a number of severe shortcomings. Retrieval is restricted to superficial representation patterns, e.g., strings or graphical shapes. Apart from that, machine retrieval of the represented objects, i.e., the content, is not possible because these objects – as well as corresponding types or classes – simply do not exist in the presentation software. Reuse of graphical elements is restricted to copy and paste, causing an ever growing amount of redundant material. In addition to that, integrity is jeopardised by the lack of semantics: There is no protection against absurd or contradictory content. Independent from semantic integrity, there is no support for a coherent representation of content within an organisation. This is the case both for concepts and graphical layout. The lack of semantics also limits the use of machine analysis and transformation. Finally, the lack of a conceptual foundation prevents integrating the content of a presentation with data that reside in an information system. It is only possible to copy representations of data, such as strings, into the presentation, which will, however, cause the loss of the original semantics. If, for instance, a pie chart is used to represent the revenues of a set of branch offices, this information is lost inside the presentation software. The implications of the poor conceptual foundation are remarkable. While the plethora of presentation slides that exist in many companies could serve as a versatile knowledge repository that is tightly integrated with the corporate information system, they are not much more than an amorphous accumulation of textual and graphical symbols. As a consequence, there is only little protection of the investment into the creation of presentation slides.

This paper presents an approach to overcome the dissatisfactory production and use of presentation slides. The approach puts specific emphasis on reuse and integrity. For this purpose, it suggests the use of domain-specific modelling languages (DSML) and corresponding model editors to develop slides that support methodical analysis and that promote a higher level of reuse and integrity. The proposed solution is not intended as a total replacement of today's presentations. Instead, it is aimed at augmenting traditional content with conceptually grounded graphics. For this purpose, it provides a conceptual foundation for integrating various types of content and for storing presentations in a common repository. While the approach targets mainly business presentations, it should be applicable to a wide range of other presentations, too. On the one hand, the approach is motivated by our experience with the production and use of presentations for teaching purposes. We often copy graphics produced by modelling tools into presentations. This does not

only slow down the production process, it also removes the semantics originally embedded in the diagrams. On the other hand, the approach is inspired by our work on enterprise modelling in general and on the specification of domain-specific modelling languages in particular.

The paper starts with a requirements analysis. Against this background, the conceptual foundation of the proposed system is developed. Subsequently, the corresponding tool is presented. Since the use of the tool requires changing existing patterns of producing presentations, organisational guidelines for establishing a corresponding practice are suggested in the following section. A discussion of related work is followed by an evaluation of the proposed solution. While the research presented in this paper corresponds clearly to the so called Design Science approach, we did not take the corresponding method (Hevner et al. 2004) as a model. Instead, we refer to an approach for the configuration of research methods (Frank 2006b). Different from Hevner et al. it does not prescribe the use of behaviourist approaches to evaluate an artefact. Instead, it emphasises the need for transparency of underlying assumptions and the use of multiple approaches to justifying requirements and solutions.

2 Requirements

To develop a foundation for more sophisticated presentation development and management tools we distinguish a macro and a micro view on creating and using presentations. The macro view focuses on the creation entire presentations. It should not only promote reuse, but also foster integrity. Different from creating isolated presentations, it should be possible to store and access the parts of a presentation such as slides and content of slides. The micro view – which is of higher relevance for the objective of the paper – focuses on the creation of graphical representations that are supposed to guide the analysis of complex subjects. For example: a presentation that aims at assessing a firm's current IT strategy and showing options for shaping the future IT strategy. Accomplishing this task requires knowledge to structure the domain in a purposeful way. In addition to that it requires knowledge about the process of developing and structuring a business presentation. Since graphical visualisation can promote comprehensibility, it needs to be decided how a graphical representation should look like. This recommends accounting for the expectations and skills of the prospective audience. Since business analysis is often not restricted to concepts but also includes considering instance-level data, the required data need to be integrated somehow from the respective sources. For instance: After associating an ERP system with the business process types it supports, one could include the total annual revenues generated through the corresponding process instances.

Today, successful action in both, the macro and the micro view, is substantially determined by the limitations of prevalent technologies, such as file systems and presentation tools. The following requirements for presentation development and management systems (PDMS) that support the creation and use of presentations more effectively than current tools respond to the macro and the micro view. Note that the requirements are not meant to be complete. Instead they are intended to focus on essential aspects.

Requirement 1: A PDMS should include a conceptual foundation that provides developers and users of business presentations with concepts to structure and analyse the domain of interest (focus on micro view). Rationale: Structuring a domain appropriately is of pivotal relevance for professional analysis. At the same time, it is a demanding task that overburdens many users. For instance: concepts such as 'strategy', 'IT resource', 'IT architecture', 'IT costs', 'business process' would help with preparing for analysing and redesigning an IT strategy. Note that this means to specify the semantics of these concepts.

Requirement 2: For various reasons, graphical representations are of pivotal relevance for presentations. Hence, users of a PDMS should be supported with the creation of diagrams that are consistent with respect to a common terminology and the graphical notation. For this purpose, users should be provided with some sort of reusable concepts and related graphical symbols (micro view). Rationale: For creating meaningful diagrams that correspond to organisational standards, it is important to provide users with effective guidance. Reusable concepts and graphical symbols do not only promote productivity, but contribute to quality and integrity, too.

Requirement 3: A PDMS should support interactive views (micro view). Rationale: For a graphical representation to serve as a powerful instrument for analysis, a corresponding tool should provide meaningful operations, e.g., to perform calculations, to change the level of detail or to navigate to associated representations.

Requirement 4: It should be possible to integrate data from other systems (macro and micro view). Rationale: If operational level data are required for presentations, they are usually copied to a slide which results in the loss of the original semantics. To support timeliness and integrity, it would be better, if the semantics of data was preserved, i.e., if the presentation software was integrated with the systems that provide the data.

Requirement 5: A presentation should allow for including any kind of content – not only graphical diagrams. This includes text, tables, graphics, etc. (macro view). Rationale: The wide range of purposes and constraints to be accounted for with the creation and use of presentations demands for a versatile, flexible approach that should not restrict content to a specific type.

Requirement 6: It should be possible to create a presentation by reusing composable elements that are stored and maintained in a common repository. In order to promote reuse the composable elements should cover various levels of detail. They should be directly accessible in the repository (macro view). Rationale: Only if a presentation is decomposed in (widely) self-contained elements, its content can be reused in a consistent way – in an ideal case by defining references to the elements in the repository only. Otherwise there is need for synchronisation mechanisms.

Requirement 7: Content should be widely independent of its presentation (macro and micro view). Rationale: The adequate presentation of content depends on the specific context, which is, among other things, characterised by the targeted audience and corporate standards. Separating content and presentation allows for (re-)using content in different contexts.

Requirement 8: The system should provide for sophisticated retrieval (macro and micro view). Rationale: A presentation repository can be expected to grow to a remarkable size. Therefore, finding adequate elements can be a challenge that threatens the utility of the entire approach.

Requirement 9: It should be possible to extend/specialise the conceptual foundation of graphical representations. Hence, there should be mechanisms that allow for convenient and safe adaptations (micro view). Rationale: With respect to the huge variety of domains and topics, it would not be reasonable to expect the conceptual foundation of a system to be complete.

Requirement 10: A PDMS should guide users with the creation of a business presentation. This includes the creation process as well as the overall structure of a presentation (macro view). Rationale: Developing a business presentation can be a demanding task. Therefore, providing guidelines for organising the process, e.g., through a process model, and for structuring a presentation, e.g., through pro-

totypical patterns, promises to promote productivity and quality.

3 Conceptual Foundation

At its core, the proposed solution is based on the idea to replace certain drawings in presentations with conceptual diagrams that are created with domain-specific modelling languages. Usually, a conceptual diagram is a view of a corresponding conceptual model. Different from a model, it is characterised by a specific layout. A conceptual diagram may also integrate views on more than one model. For instance: A diagram may represent a business process model and parts of an associated model of IT resources. For creating conceptual models and related diagrams, the use of domain-specific modelling languages (DSML) is suggested. Conceptual diagrams allow for enriching graphical representations with formal semantics, to foster integrity and to provide a foundation for meaningful user interactions.
Therefore, we refer to slides that contain conceptual diagrams as 'smart slides' and call a presentation that includes smart slides 'smart presentation'. Hence, the main focus of this paper is on a conception of conceptual diagrams with respect to requirements 1 to 4, 7 and 9. However, a conception of conceptual diagrams is not sufficient. Instead, there is need for integrating conceptual diagrams with presentations that may also include other types of content (requirement 5) and that satisfy the demand for reuse (requirements 6, 7). Therefore, the illustration of the targeted solution starts with an outline of a compound architecture that provides the context for using interactive conceptual diagrams within a presentation.

3.1 Compound Architecture

The object model depicted in Fig. 1 illustrates the proposed compound architecture. A presentation consists of an ordered collection of references to slides – which in turn refer to the frames they include. There are master frames, such as headlines, footlines or logos as well as regular frames. A regular frame is an instance of a frame class, such as `Text`, `Graphics` or – more specific – `Question`, `Assignment`, `Citation` etc. It is represented as a rectangular area on a slide. Frames may overlap. Its default relative size is defined by respective attributes in the class AbstractFrame. The attributes in the class FramePosition serve to define its concrete position and its possible enlargement within a particular slide. Frames may overlap. The layout of slides is defined in separate objects. On the highest level, an object of the class `PresentationStyle` can be used to specify the layout style – such as default font, background colour etc. – for an entire presentation. The styles of chapters or slides and of particular frame classes can be defined in further objects. This is indicated through the classes `ChapterStyle`, `SlideStyle`, `QuestionFrameStyle` and `AssignmentFrameStyle`. In case, there is no particular style for a chapter, the corresponding values are obtained from the associated object of the class `PresentationStyle`. The same pattern applies to objects that serve to define the style of slides. They obtain the corresponding values from an object of the class `ChapterStyle`, if they are not used to define a deviating style. Note that the relative position of head- and footlines will usually be defined in respective style objects. Objects of the class `FramePosition` may be used to override the default in exceptional cases.

The object model shown in Fig. 1 illustrates how a presentation can be constructed through references to elements in the repository and how the style of a presentation can be defined on various levels of detail. To include data from other systems, access to these data can be specified in an instance of the class `Interface`. The integration of conceptual diagrams is indicated by the highlighted classes at bottom right of the class diagram. A diagram is assigned to one slide, i.e., it cannot be distributed over various slides. In case, it is too large for fitting one slide, it would

be required to split the diagram into several diagrams, each of which could be assigned to a slide. If the modelling language provides for decomposition, the corresponding concepts could be used to decompose a large diagram into a set of smaller ones. Alternatively, a large diagram could be assigned to a single slide, if the corresponding tool provided a zooming function – which is the case for the tool presented in Sect. 4.

Composing presentations as ordered collections of references to slides will clearly help to avoid redundancy, since a set of presentations may share slides that reside in the repository. As a consequence, changing a slide in the repository will result in a simultaneous and consistent update of all involved presentations. This is a clear advantage especially in those cases, when obvious errors were found in slides. However, the issue of modifying slides in the repository needs to be considered in more detail.

First, a presentation may be regarded as a document that should preserve its state. In this case, keeping references to slides in the repository that might be subject of change would be harmful. To cope with this requirement, a 'deep' copy of a presentation – in the sense of a snapshot (value semantics) that preserves all values (slides and frames) – could be stored in an additional database that would mainly serve documentation purposes. Second, there is the case of partially modifying a slide in the repository. This would be handled by copying the slide together with its references to the included frames. Subsequently, the references of those frames that were to be changed would be replaced by references to modified or newly created frames. If only minor changes are applied to a slide, e.g., one out of several frames get replaced, the remaining commonalities will be redundant and may thus threaten the integrity of the repository. The proposed architecture accounts for this problem with a simple conception of variant: It allows for defining a slide as a variant of an existing slide, which in turn may be a variant of a further slide. In the object model in Fig. 1 this is indicated through the association 'variant of' with `RegularSlide`. It is supplemented by a constraint to prevent cyclic associations, which is not shown in Fig. 5. Note that such a simple concept of variant does not allow for automatic updating of variants. Instead, it supports detecting variants that may be affected when a particular slide has been changed. The actual modification of the variants would then require human action. Defining a more elaborate semantics of slide variant would require referring to the set of included frames and the respective content. While this is not at the focus of the paper, we will get back to this issue in Sect. 5 with respect to the modification of models and diagrams.

In the ideal case, the elements a presentation is comprised of are loaded from the repository during the execution of a presentation. If the required connection is not available, the elements can be copied to the machine that runs the presentation. If they are modified, they may be checked in to the repository later – which requires an appropriate synchronisation protocol. Note that the object model in Fig. 1 is not intended to be comprehensive. It is restricted to those classes that are required for representing compound presentations. Further classes (e.g., for animation purposes), methods and constraints have been widely omitted. Also, it does not include management classes for, e.g., inserting or deleting slides, printing slides, editing text and graphics etc. While the remarkable complexity of the architecture needs to be hidden from users, the features it enables should be made available to users nevertheless. This requires putting special emphasis on the design of the user interface. The screenshot in Fig. 5 gives an idea of how this could be accomplished.

3.2 Language Architecture

Different from current presentation tools, a graphical representation would not be restricted to drawings or imported graphics in various formats,

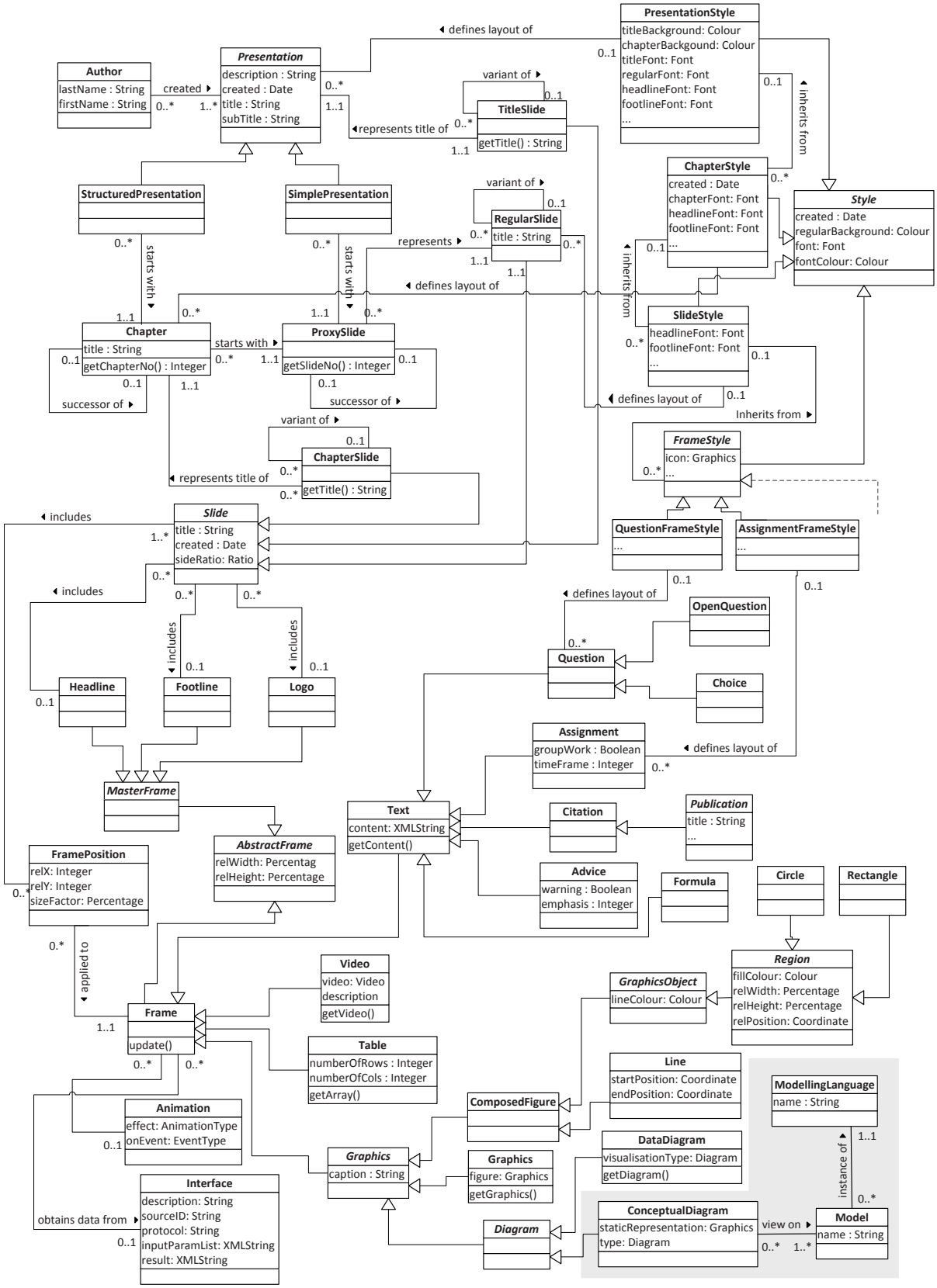

Figure 1: Object Model of Component Architecture

but would also allow for creating conceptual diagrams. To foster productivity, the presented approach emphasises reuse on two different levels. At first, it proposes to use domain-specific modelling languages (DSML) for creating conceptual models. A DSML provides users with modelling concepts that represent a technical terminology. Therefore, the user does not have to (re-) construct domain level concepts, e.g., 'business process', 'task', 'role', 'objective' etc. on his own by using generic concepts such as 'class', 'attribute' etc. Second, models that reside in the repository as a reference, can be reused – and adapted using the respective DSML. The use of a DSML also fosters comprehensibility by featuring a specific graphical notation. Last but not least, a DSML promotes integrity, since its syntax and semantics prevent inappropriate models more effectively than a general purpose modelling language (GPML) like the UML. Figure 2 illustrates the advantages of DSML: The class diagram on the right side is perfectly valid, both with respect to its syntax and semantics, because within the GPML there is no differentiation between classes. Different from that, the DSML provides concepts that represent the domain-level terms 'Server' and 'ERP'. Therefore, it does not allow for expressing that a server runs on an ERP system. Note that the excerpts both of models and metamodels are substantially simplified. Also, the levels of abstraction do not exactly correspond.

With one or more built-in DSML editors, a PDMS features functions that are similar to those of a modelling tool. It allows for creating and modifying conceptual diagrams that can be interacted with during a presentation: to apply changes, to decompose elements of a model, to perform calculations or to access related data.

The conception of smart slides that is proposed here was inspired by a method for multi-perspective enterprise modelling (MEMO) (Frank 2002; Frank and Lange 2007). Enterprise models are models that integrate models of the information system, such as object models, component models etc., with models of the organisational action

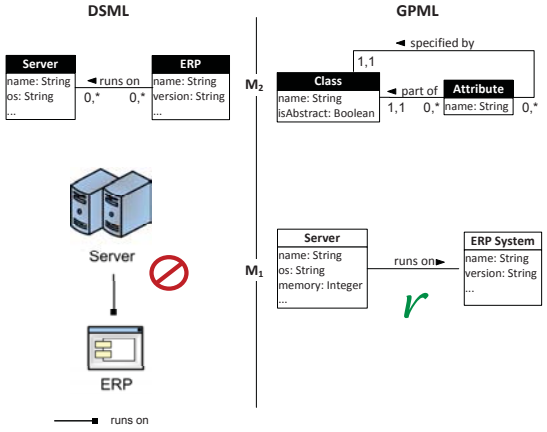

Figure 2: Illustration of integrity gain through DSML

system, e.g., business process models, strategy models etc. Enterprise models are usually specified with DSML. The semantics of DSML allows for various forms of machine analysis and provides a foundation for systems design. At the same time, an enterprise model serves as a medium to foster communication between stakeholders with different perspectives.

Often, DSML are specified by metamodels. Metamodels are especially suited for this purpose for two reasons. First, they provide an advantageous foundation for developing corresponding model editors because they can be transformed to object models in a straightforward manner. Second, they enable language specifications that are easier to comprehend for many than grammars.

In order to foster integration and extensibility of languages, the above-mentioned method features a language architecture. It is based on a common meta metamodel (Frank 2011) that specifies the abstract syntax and semantics of the MEMO metamodelling language. The metamodelling language allows for including OCL statements to refine language semantics. The meta metamodel is instantiated into the metamodels which specify the abstract syntax and semantics of various DSML. The conceptual foundation of smart slides is built on a corresponding language architecture

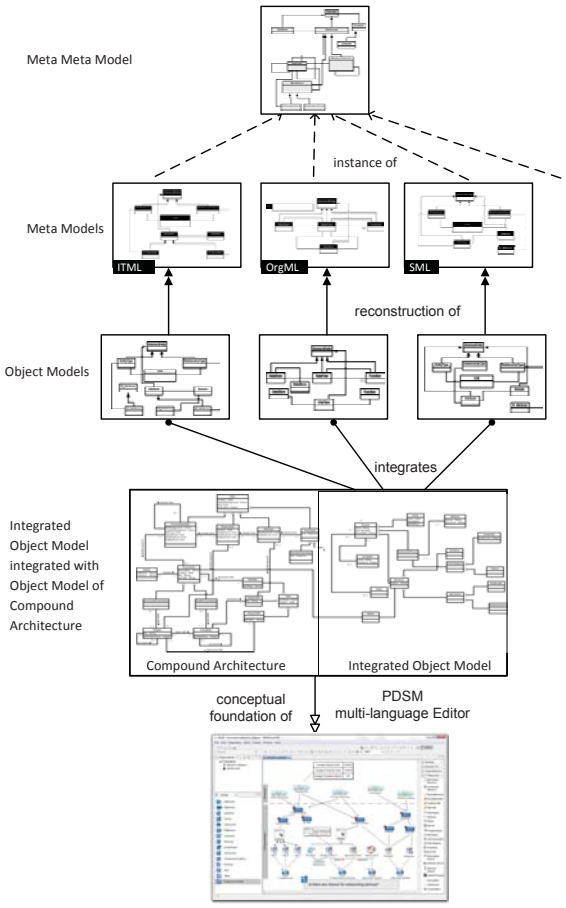

Figure 3: Language architecture and conceptual foundation of PDMS

– and makes use of existing languages for enterprise modelling. Currently, these DSML reflect the requirements of enterprise modelling. They include a language for modelling organisations, both organisational structures and business processes, OrgML, a language for representing strategic aspects such as goal systems or value chains, SML (Frank and Lange 2007), and a language for modelling IT resources on various levels of detail, ITML (Kirchner 2008). Further languages target modelling of resources (Jung 2007) or various aspects of corporate knowledge management (Schauer 2008). Distinguishing these languages is mainly motivated by the need for reducing complexity: While it is conceivable to define one multi-purpose language that allows for creating all intended diagram types, such an approach would result in a level of complexity that could hardly be managed anymore. To provide for the development of model editors, the various metamodels are reconstructed as object models. The object models represent the language specification and cover further aspects that are required for the implementation of a modelling tool such as time stamps, access rights etc. As will be outlined in Sect. 5, the object models are generated to a large extent by a metamodel editor that serves to specify the metamodels. In order to support the integration of modelling languages, the object models are integrated to a common object model which serves as the conceptual foundation of a multi-language model editor (see Fig 3). It is integrated with the object model that represents the component architecture (Fig. 1) to form the conceptual foundation of the PDMS (Fig 3). Whenever a user adds a frame of the class `ConceptualDiagram`, a set of model editors will be provided that serve to create an instance of the frame class which is part of the overall presentation (illustrated in Fig. 6).

The integration of models is accomplished by integrating the respective metamodels, i.e., through common concepts shared by the corresponding modelling languages. For instance: To integrate the IT resource model depicted in the screenshot in Fig. 5 with a business process modelling language both languages need to share concepts such as 'business process' and 'IT resource'. The architecture allows for adding further languages by including respective metamodels. The creation of corresponding model editors is supported by the tool environment (see Sect. 5).

3.3 Exemplary DSML

Currently, the method includes modelling languages which were designed for creating enterprise models. While some of these, e.g., languages to create object or component models, are suited for very specific technical presentations only, others, such as organisation, resource or strategy modelling languages offer concepts and graphical notations that seem appropriate

for a wide range of applications. A diagram can integrate views on models specified in different modelling languages. For instance: the business process types represented in a business process map can be associated with corresponding goals represented in a strategy net (Frank and Lange 2007), or with classes in a class diagram or artefacts in an IT resource diagram (Frank et al. 2009). The excerpt shown in Fig. 4 illustrates how the integration of modelling languages provides the foundation for multi-language diagrams as the one shown in Fig. 5. It is specified with the MEMO MML, which features a specific graphical notation to promote a clear distinction of object models and metamodels. Note that the shown metamodel is a substantial simplification of the actual metamodels. It includes only a small subset of meta types – especially the concepts for modelling business processes are reduced to a minimum – and omits multiplicities as well as OCL constraints.

The excerpt includes concepts needed for representing abstractions that are prevalent in consulting firms such as the balanced scorecard or value chains on the strategy level. These concepts are associated with further concepts provided by other modelling languages, e.g., `BusinessProcess`, `OrganisationalUnit` or `SoftwareService`. Note that the example diagram types referred to in Fig. 4 name only a few. Also, since the concepts included in the given set of modelling languages can be combined to serve more specific purposes. Nevertheless, further modelling languages are required to cover the range of topics and purposes addressed by presentations in the realm of business and information systems. The graphical notations of the present DSML were created by a graphic artist.

Conceptual models focus on type-level data. This is for a good reason: Analysis should emphasise essential aspects and should not be distracted by instance-level peculiarities. However, sometimes there is demand for including data that refer to instance states. This is especially the case, if a type includes features that reflect an aggregation of corresponding instance values. For example: A business process type may be characterised by the average execution time of its instances during a certain time frame. An organisational unit may have a feature that represents the average salary of all included positions, etc. To promote timeliness, integrity and productivity, instance level data should be obtained from those systems that manage them. The metamodelling language accounts for this aspect by providing two specific concepts (see Frank 2011): 'Intrinsic' features allow for specifying features of a meta type that are supposed to be initialised on the instance level only (and not on the type level). For example: The meta type `Indicator` includes the attribute `value`, which is not a feature of an indicator type, but is only to be initialised for a particular instance. Features may also be marked as 'obtainable' (see, e.g., `averageCost` within `BusinessProcess` in Fig. 4) to indicate that the corresponding values might be obtained from other applications. This could be accomplished through an interface class like the one depicted in Fig 1. However, a more sophisticated integration would require a different approach (see Sect. 7).

4 Tool Environment

The proposed architecture and prototypical implementation of a PDMS is based on an existing modelling environment[1]. It was developed with the Eclipse Modelling Framework (EMF) and the Eclipse Graphical Modelling Framework (GMF). The implementation of the component architecture and the corresponding editing features is only prototypical. With respect to the standard set by current presentation tools, implementing these features would require an enormous effort – at the same time there is hardly need for a proof of concept. The main purpose of the prototype is to demonstrate two features: the integration of a multi-language diagram editor with a presentation tool and the support for adding further model editors.

[1] available for download at `http://www.wi-inf.uni-duisburg-essen.de/FGFrank/download/memo/`

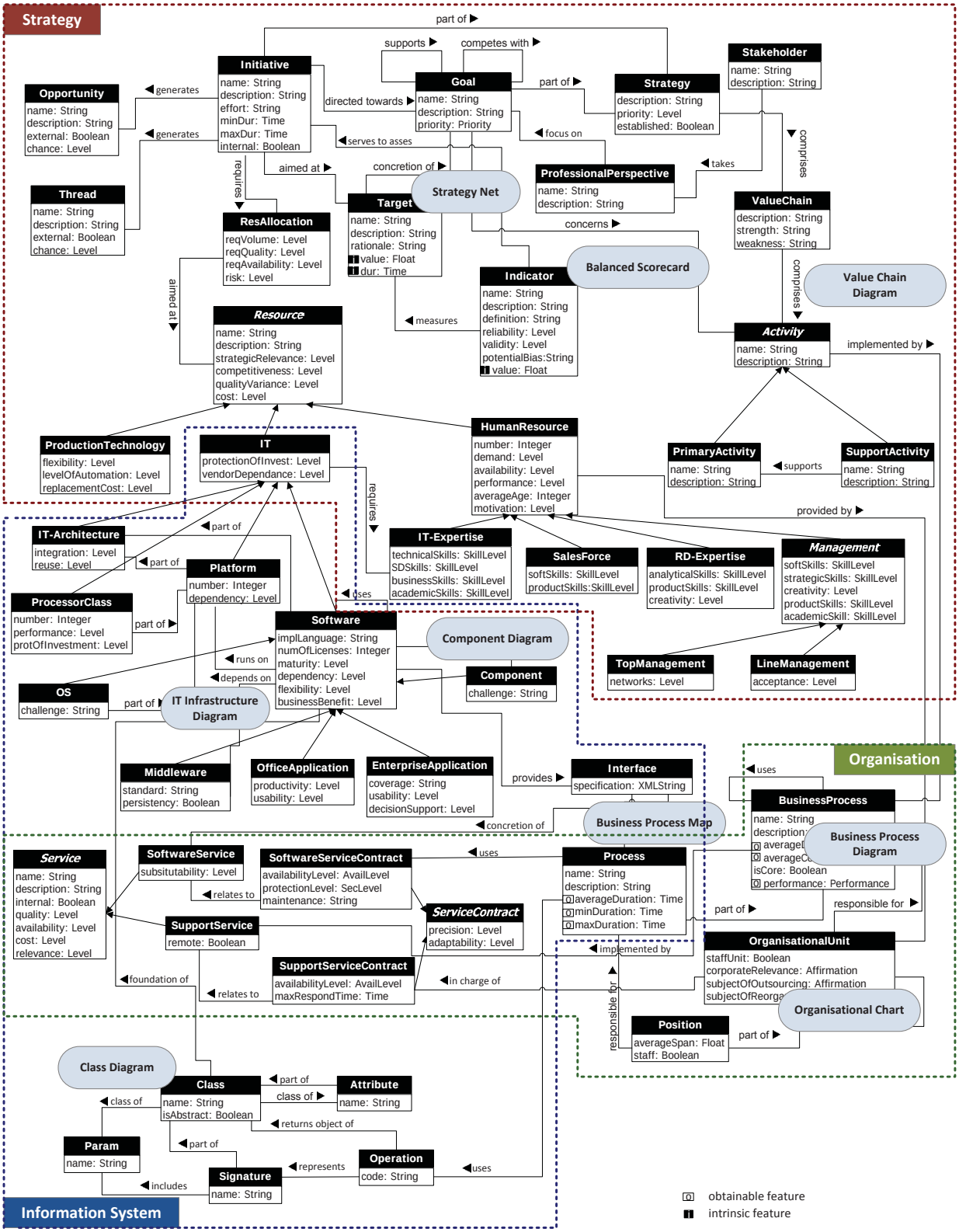

Figure 4: Excerpt of integrated metamodels

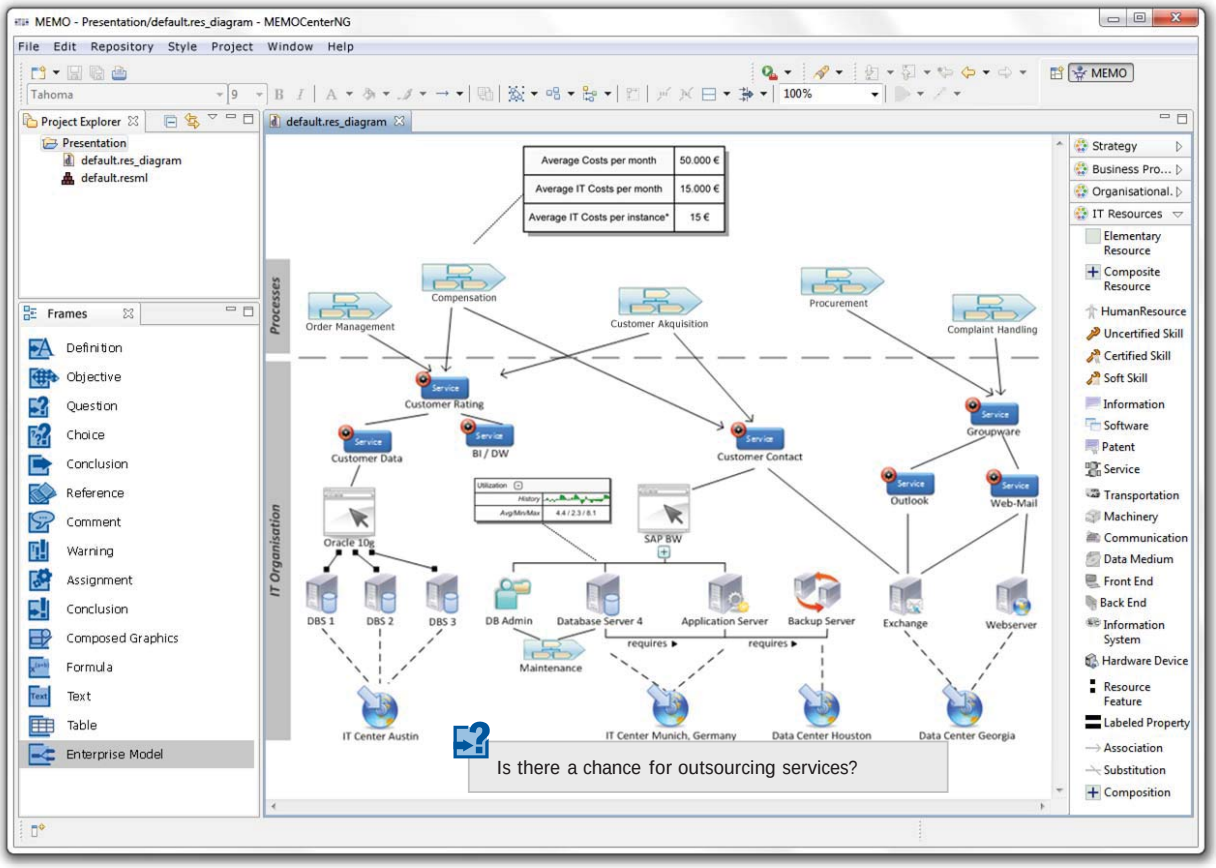

Figure 5: Screenshot of PDMS editor

Using a PDMS is similar to using a traditional presentation tool. This is mainly for the reason that a presentation is still regarded as a series of slides. However, the use of a PDMS is different in two respects: It puts emphasis on typed content and it promotes integrity and reuse by using a common repository. The latter aspect demands for establishing a corresponding work practice – in other words: for reorganising the preparation, creation and maintenance of presentations. Respective guidelines will be presented in Sect. 5. A presentation is built from scratch by defining one slide after the other. Defining a slide means either reusing an existing one or creating a new one. If the user finds an appropriate slide in the repository, he will establish a reference to this slide. If he wants to modify this slide, he can replace the included frames by new or modified ones. If the modified slide has a common core with the original one, he can define it as a variant (see Sect. 3.1). If there is no slide in the repository to start with, a new slide would be created. Each new slide can be regarded as an empty canvas. It is filled by dragging a typed frame from the tool palette (e.g., text frame, a question frame, a conceptual diagram frame etc.).

Each frame type is associated with a corresponding editor. The tool palette is shown on the left side of the screenshot in Fig. 5. If a conceptual diagram frame was selected, a context-specific tool palette will pop up that provides the user with a set of modelling languages to choose from (right side of the screenshot in Fig. 5). Selecting a model editor provides the user with various choices. He can select an existing diagram – which provides a view on an existing model. Al-

ternatively, he may copy and modify an existing diagram. He also may create a new diagram of an existing model. It is also possible to create a new model and a corresponding diagram. Finally, he may want to change the models a diagram corresponds to. This would constitute a problem only, if model elements represented in a diagram were deleted. In this case, the tool allows for creating a new version of a model, while the old version together with the related diagrams remains in the repository. In its current version, the tool environment does not provide support for defining and managing variants of models, since the specification of a sophisticated concept of model variant with all variants sharing a common core is subject of ongoing research. The only option is to explicitly define a model as a variant of an existing one. Third, modelling languages may be modified, too. Currently, there is no mechanism implemented in the tool that would support merging a model and its diagrams to a newer version of the corresponding languages. Hence, the modification of languages recommends keeping older versions and their models. Currently, the repository manager is realised through persistency services provided by the EMF. Models are stored as XML files in a structure that is defined by the so called 'notation meta-model' of the GMF.

Note that the tool allows for creating multi-language diagrams by combining multiple DSML. Also, traditional frame types – such as text or drawings – can be combined with conceptual diagrams on one slide. In the example shown in Fig. 5, a business process modelling language is supplemented by a language for modelling IT resources. The frame that contains the conceptual diagram is supplemented by an instance of the frame type `OpenQuestion`. Also, model elements are associated with corresponding data on the instance level, in this case data from accounting, which may e.g., originate in an ERP system. In addition to creating and editing diagrams, the model editors allow for running analyses – e.g., detecting the number of business process types that are supported by a certain IT resource, or calculating the average span of control of an organisational structure.

The elements of a diagram may be associated with model elements that are not part of the diagram. For instance: The business process types shown in the example in Fig. 6 might be specified in corresponding business process models. The organisational role 'DB Admin' may be specified in an accompanying elaborate model of the organisation structure. With respect to including a diagram into a presentation, this implies a number of challenges. First, data that is referred to in a diagram, e.g., the average cost of a business process of a certain type, may change over time. To cope with this problem, the user could decide whether to copy obtained references or values into a diagram. Second, the layout of a diagram that is referenced by many presentations may be changed. In the easiest case all references are kept as they are, resulting in a modified diagram layout within all affected presentation. If it is required for some presentations to preserve the layout of the diagram, a variant of the respective slide (see Sect. 3.1) could be created that would contain the original state of the diagram – which would still refer to the same model.

A DSML may require modifications over time. Also, there may be need to add further DSML. To cope with this demand, the tool environment also includes a component that supports the development of model editors. It reflects the language architecture shown in Fig. 3. The metamodel editor uses the metamodelling language specified by the meta-metamodel for creating and modifying metamodels. As soon as a modelling language is created or modified, the editor transforms it into a corresponding Ecore instance, which serves to represent metamodels within the GMF. This includes the transformation of OCL statements. Subsequently, further specifications, such as the concrete syntax, have to be added. This still requires remarkable expertise and effort. However, the MEMO metamodel editor and the GMF, it

is built on, facilitate the construction of additional model editors to a large extent. After a new model editor has been completed, it is integrated with existing editors, mainly be integrating the object model, it is based on, with those of other modelling language editors. Thereby, single language editors become part of a multi-language editor that allows for creating diagrams that represent models specified by different languages. Figure 6 illustrates this process. For more details see Frank et al. (2009). Nevertheless, code generation implies a well-known problem: Whenever a metamodel is modified, generating code threatens to destroy manual enhancements of existing code. To counter this problem, the generated code is separated by clear rules from code that is added later on. However, this measure does not cover all possible modifications.

The integration of additional model editors with existing ones requires language concepts that are already part of existing languages. For instance: A new DSML that serves to model human resources is specified in the metamodel editor. To allow for associating a business process model with a model of human resources, the language specification needs to include an adequate concept of the business process modelling language – et vice versa. This will require recompiling the respective editors.

The larger a repository of smart slides, the more important it becomes to provide for retrieval. Retrieval of diagrams can be based on the included concepts, e.g., 'indicator', 'primary activity', or designators, e.g., 'order management', 'sales' etc. Furthermore, retrieval is supported by a faceted classification that is based on the high-level framework of the underlying method for enterprise modelling. It structures an enterprise into three generic perspectives – strategy, organisation and information system, each of which can be further differentiated into four aspects – resources, process, structure and goals.

Each of the resulting 12 foci can be further divided into a set of customisable professional perspectives, e.g., financial, sales etc. This high-level

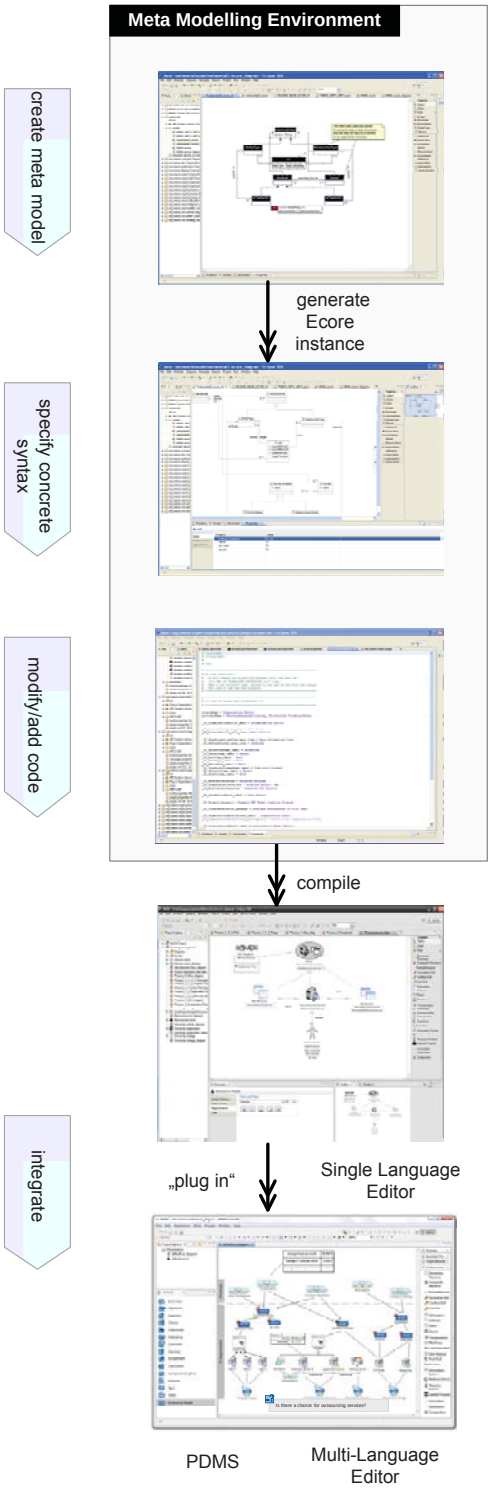

Figure 6: Workflow for developing and integrating additional model editors

scheme – called 'topic map' (see Fig. 7) – serves to classify more specific topics, e.g., 'process improvement', 'assessment of IT architecture' etc. Each presentation or slide can be assigned any number of topics.

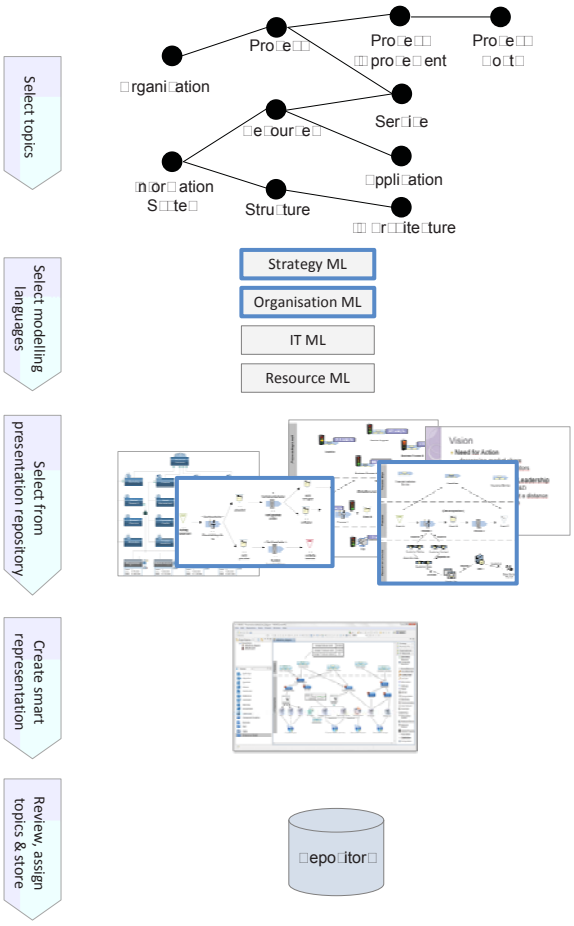

Figure 7: Process model for creating smart presentations

5 Organisational Guidelines

The conception of slides as conceptual models revaluates business presentations and makes them a core repository and medium of organisational knowledge. DSML promote reuse and foster integrity and coherence of the represented knowledge. They also reduce complexity by providing proven concepts for purposefully structuring a domain. However, often, the reduction of complexity implies increasing it at first – in terms of acquiring new concepts, tools and procedures. This is the case with sophisticated presentations, too. First, the specification of a DSML is a remarkable challenge caused by the specific complexity that rises from the construction of powerful abstractions. Second, the use of a DSML and the reuse of existing models demand for specific skills and for professional training. Prospective users need to understand the core ideas and benefits of using a DSML instead of merely drawing graphical representations. Finally, it is required to reorganise the construction, use and maintenance of presentations. This includes the process of creating and (re-)using business presentations as well as the professional roles that are required for exploiting the potential offered by PDMS.

Creating a smart presentation is mainly a conceptual challenge. We call the corresponding role *scriptwriter*. It requires creating and structuring arguments for the targeted audience with respect to a certain objective. At the same time, the required skills in graphical design are less demanding, because a DSML includes a graphical notation already. A scriptwriter can either create a presentation from scratch or modify an existing one. *Moderator* is a further role that is responsible for mediating a business presentation and for involving the audience in the interaction with smart slides. To support the idea of a steadily growing knowledge base of smart business presentations, the role of a *reviewer* is required to care for quality assurance. This does not only include the quality of presentations per se, but also the overall consistency and coherence. Ideally, it requires an outstanding qualification with respect to the entire scope of the represented knowledge and the DSML in use. Depending on the size of the knowledge base and the quality produced by the scriptwriters it may also require a substantial effort. How much an organisation is willing to spend for reviewing and revising smart presentations depends on the benefit to be expected from an elaborate knowledge base. For those organisations that regard the creation of business presentations as a core competence, e.g., large consultancy firms, the roles of a *language*

designer, a *graphical notation designer* and a *tool administrator* may be required. A language designer creates the conception or modification of DSML. For this purpose, he specifies or changes metamodels using the metamodel editor. In cooperation with the graphical notation designer he is in charge of the graphical notation. In addition to that, his obligations include the specification of particular requirements for the corresponding model editor. The tool developer extends the basic model editor that is generated by the tool environment according to the specific requirements. Language design and tool development require highly specialised skills. Therefore, it will often be appropriate to locate these roles with specialised service providers. Those could not only develop and maintain DSML for creating business presentations, but also provide reference presentations that can be adapted to specific needs.

The process model that is depicted in Fig. 7 illustrates the design, reuse and management of smart presentations in an organisational setting. At first, a scriptwriter, who wants to create a new presentation would start with searching the library of existing presentations. For this purpose, he checks the available topic map for topics that fit the intended presentation. These topics refer to available presentations, slides, frames and modelling languages. He would then either reuse existing slides or create new ones from scratch (see Sect. 4).

After a presentation is completed, it is stored in the repository. This can either happen directly or after it had been approved by the reviewer. Any task within this process may produce requirements for the development of DSML and tools. They should be gathered and revised by the reviewer. If a modification of an existing modelling language is regarded as necessary or even a new modelling language is required, a language designer would use the metamodel-editor to modify/create the corresponding metamodel, which would then – according to the workflow shown in Fig. 6 – be transformed step by step to a new instance of a corresponding model editor that finally would be integrated with the PDMS. If during the creation of a presentation new topics occur, the reviewer is supposed to update the topic map.

6 Related Work

To the best of our knowledge, there is no work that is directly related to the presented approach. However, there are various streams of research that address certain aspects. In Tantau (2007) a LaTeX class is presented that is based on a conception of presentation which shows similarities to the approach suggested here. It defines a structure of presentations that includes concepts to differentiate content, such as graphics, quotations or questions. Despite the power of LaTeX, it is not intended for creating interactive presentations, but documents. Among others, this is illustrated through a concept called 'frame' that is clearly different from the one proposed for smart slides: A frame in the corresponding LaTeX class consists of a sequence of slides. It serves to 'simulate' animations within the limitations of PDF viewers. With respect to the support for analysis, PDMS resemble decision support systems (DSS). DSS are sometimes accompanied by the demand to account for different 'cognitive styles' of prospective users with particular emphasis on graphical representations. Smart presentations can serve different cognitive styles, since the use of DSML allows for stressing different perspectives on a subject, e.g., a business process perspective or a resource perspective. Different from DSS, the current prototypical implementation does not include specific decision models. Those could, however, be added. A more detailed comparison with DSS is problematic, since DSS represent a class of systems that is rather characterised by its purpose than by its conceptual foundation. This is slightly different with expert systems (XPS) which usually, but not necessarily, are characterised by a declarative representation of knowledge. While graphical representations

are not at the core of XPS, they allow for monotonic extensions and deduction. These are both valuable features with respect to maintaining and evaluating knowledge. Unfortunately, modifications of conceptual models are not guaranteed to be monotonic. Also, the MEMO metamodelling language does not, apart from the use of specialisation, support deduction. Certain aspects of PDMS relate to various kinds of further knowledge management systems (for an overview see Maier 2004), such as integration with operational level systems (through language concepts that correspond to implementation level concepts), collaboration (smart slides as a medium to foster communication in heterogeneous groups) and support for cognitive processes (since a DSML is supposed to represent carefully designed and well suited linguistic structures).

'Business Intelligence' (BI) systems provide features that allow for extracting data from various sources and for visualizing them through graphical representations. In recent years, so called management dashboards have gained remarkable attention (Eckerson 2005; Few 2006). They are supposed to provide versatile graphical frontends to enterprise software such as ERP systems or data warehouse systems. Their purpose is similar to that of smart slides in two respects. First, they are supposed to reduce complexity by focusing on essential aspects of a subject. Second, they emphasise a graphical representation. However, dashboards – or graphical representations generated by BI systems in general – usually lack a conceptual foundation that accounts for the semantics of the represented knowledge. Also, their focus is more on (aggregated) data – sometimes real-time. Note, however, that DSML are well suited to develop dashboards that go beyond mere visualisation tools (Frank et al. 2009). For instance: A graphical representation of the IT infrastructure may be supplemented with data on costs, availability etc. If the representation is realised by using a DSML, the corresponding diagram can be modified, e.g., by adding further resources, and the effect of a modification can possibly be analysed by the corresponding software.

Research on domain-specific ontologies is similar to the development of DSML in the sense that a domain-specific ontology is focusing on reconstructing technical languages, too. However, there are a few clear differences. Usually, ontologies are not embedded in a language architecture. There is usually no clear differentiation of language and language application. While this promotes flexibility, it poses a problem with respect to building model editors, since they require a separation of meta language and language. Second, ontologies typically do not account for a graphical representation. Also, business ontologies such as Andersson et al. (2006), Andersson et al. (2009), Gordijn (2004), Osterwalder (2004) usually focus on a more abstract level, stressing basic terms of the targeted domain – such as 'event', 'feature', 'resource' (see, for instance, Andersson et al. 2009). While this extends the range of possible (re-)use, it implies a higher effort for modelling a particular domain, since many terms still need to be reconstructed from the basic concepts of the ontology. The logic-based languages, ontologies are usually defined with, have the advantage that they allow for deduction. Unfortunately, their semantics – e.g., of generalisation/specialisation – is different from that of prevalent implementation languages. Nevertheless, a domain-specific ontology can support the construction of a corresponding DSML.

Authoring tools that support the creation of multimedia presentations can serve a similar purpose as PDMS. However, the variety of authoring tools is too large to allow for a meaningful comparison. Tools that support the creation and management of learning material also exist in a large variety. But some of them follow an approach that is similar to that of the proposed PDMS, since they feature languages to model learning content. The Instructional Management System

(IMS) initiative has produced a number of specifications for creating models of learning content[2]. However, they mainly serve to promote the realisation of interoperable learning tools. Therefore, the focus is on the specification of exchangeable content and the process of creating learning material. This includes the specification of learning scenarios. The 'languages' are specified using XML schema and lack a graphical notation. IMS does not account for specific learning subjects. Instead it stresses a more abstract level that provides concepts to describe any kind of content. The Educational Modelling Language (EML)[3] which is specified as an XML-DTD does not include concepts that relate to specific learning subjects. Instead, its focus is on concepts to describe learning material in general, such as 'learning activity', 'activity selection', 'answer-choice', 'answer-property' etc. While the EML does not account for a graphical notation, there are tools ('players') that allow for executing models of learning scenarios. There are approaches that augment IMS with a graphical notation (Laforcade 2007; Paquette et al. 2006; Sampson et al. 2006). While they foster a more convenient use of the respective concepts, they are restricted by the limits of the IMS languages.

7 Evaluation

Proposing a solution that is supposed to qualify as a research contribution requires an adequate justification. This includes both the justification of the underlying requirements and of its suitability to satisfy the requirements. The original idea was inspired by an analogy. The requirements were also influenced by this analogy and furthermore developed by analysing potential use scenarios. This is not necessarily compliant with the frequently cited principle that design science should be relevant in the sense that it addresses an actually existing problem that is given a high priority by practitioners (Hevner et al. 2004, guideline #2). While we believe that one should not ignore the manifold – and contingent – aspects of social reality, we also think that it is not a good idea to restrict scientific curiosity by the lack of problem awareness – and imagination – in practice. Hence, justifying the key elements of the proposed design is restricted to analysing their contribution to satisfy the requirements. An empirical analysis – independent of inherent epistemological problems (Frank 2006b, pp. 23f.) – is no option because the implementation and representative dissemination of a robust system is beyond the capabilities of a single research institution. Against this background, evaluating the proposed artefact is done in two steps. First, the key elements of the artefacts are compared – on a pure analytical base – against the requirements. Second, the evaluation is further differentiated with respect to contingent aspects that may influence the benefit of the proposed solution in practice. Table 1 summarises how well the presented system addresses the requirements.

The demand for integrating data from external systems (req. 4) deserves special attention, since it points at the limits of PDMS as self-contained systems. Only if PDMS are designed as integrated parts of (enterprise) information systems, they could access data of these systems without any loss of semantics. The metamodel layer provides for integrating conceptual models with data that originates in information systems: Modelling concepts can be associated with concepts to describe object models. The excerpt of the integrated metamodel shown in Fig. 4 includes one corresponding example: The meta type `Process` is associated ('uses') with the meta type `Operation`. On the model level, this would allow for associating a particular process type (e.g., 'check availability' with an operation of a certain class, e.g., 'Product.amountInStock'). In this respect, the proposed conception of a PDMS represents only an intermediate step towards a more ambitious vision: 'self-referential' information systems that are integrated with conceptual models of their own and of the context they are

[2] http://www.imsglobal.org/specifications.html

[3] http://eml.ou.nl

Table 1: Summary of evaluation

Req.		Comment
No. 1	+	The conceptual foundation is supported by the DMSL and – if available – by existing reference models.
No. 2	+	The construction of graphical representations is effectively supported by DSML that include a graphical notation. Corresponding editors guide the appropriate use of the notation elements.
No. 3	+	The integrated modelling editors allow for interaction, e.g., for navigation, composition and decomposition of model elements and for performing analyses.
No. 4	o	The metamodelling language includes concepts to prepare for integrating instance-level data from external sources. However, a tight semantic integration would require the integration of the PDMS with a corresponding (enterprise) information system.
No. 5	+	The proposed compound architecture allows for representing a wide range of content types. If required, it further concepts could be added conveniently.
No. 6	o	The conceptual foundation enables presentations that are composed as ordered collections of references to elements in a common repository. The reusable elements exist on various levels of detail. The current implementation of the repository relies on the persistency services provided by the Eclipse framework, which are not an ideal solution, since they rely on XML files.
No. 7	+	With respect to traditional content, the differentiation of frames and a corresponding presentation style – which can be defined for various scopes – enables the comprehensive separation of content and presentation. The DSML conceptual diagrams are based on a clear separation of semantics and abstract syntax on the one hand and concrete syntax (notation) on the other hand. The tool allows for conveniently changing the concrete syntax of a diagram.
No. 8	o	Retrieval of presentations and slides is supported by faceted classification based on a framework for enterprise modelling.
No. 9	o	Normally, the conceptual foundation of the compound architecture should not be subject of frequent changes. Modifying and enhancing the existing set of modelling languages is supported by a metamodel editor. However, there is need for manual interventions which challenge the integrity of maintenance activities.
No. 10	+	The use and further development of a PDMS are guided by process models that include models of respective roles.

+ satisfactory o some restrictions apply

deployed in – enabling advanced users to navigate from data and processes to graphical representations of their conceptual foundation – et vice versa (for an elaborate description of this vision see Frank and Strecker 2009). Advanced information systems of this kind could take advantage of their sophisticated conceptual foundation to provide various groups of stakeholders with diagrams – both on the conceptual and on the instance level.

Note that the evaluation omits one factor that is of pivotal importance: the quality of the DSML that are used within a PDMS. On the one hand, this requires the evaluation of the abstract syn-

tax and semantics, which faces – for serious epistemological reasons – a remarkable challenge (Frank 2006a). It also includes the assessment of the graphical notation (for a set of high-level guidelines see Moody 2009). While there has been an intensive discursive evaluation of the present DSML within the group of involved researchers, there has been only selective feedback from prospective users so far. Also, the evaluation of a DSML needs to account for its level of domain-specific semantics. The contribution of a DSML to model integrity depends on how specific it is to the targeted domain. In other words: The more domain-specific semantics is represented within a DSML, the better the chance to prevent inappropriate models. Unfortunately, this comes with a trade-off: The more specific a DSML the less is its range of reuse and, hence, its economies of scale.

So far, the evaluation was based on analytical considerations only. That is not sufficient for judging the suitability and acceptance of PDMS in practice. There are good reasons to assume that the approach will not be embraced by every user of current presentation tools. While a detailed and representative empirical investigation to address this question is – for reasons outlined above – no option at present time, a more differentiated evaluation which could also serve as a foundation for future empirical studies can be performed nevertheless. It needs to account for the contingency of economic preconditions, organisational settings and individual dispositions that influence the benefit of PDMS. From an economic point of view, the investments into a PDMS make sense only, if the benefits to be expected from increased reuse and improved integrity are sufficient to overcompensate the costs of introducing and maintaining a PDMS. This is the case for creating a common repository of slides in general (macro view), and the use of a DSML in particular (micro view). The benefit enabled by reusing existing slides and frames trends to grow with the amount of slides produced in an organisation (economies of scale) and the similarity of topics and purposes to be addressed with the corresponding presentations (prerequisite of reuse). On the other hand, the additional effort it takes to prepare a graphical representation or any other part of a presentation for reuse, will pay off the more with the amount of expected future reuse. The higher the demand for content to be correct and topical, e.g., if it is related to mission critical technical or economic aspects, the higher is the economic benefit of integrity. Also, the more integrity is regarded as a key expression of corporate performance in the sense of a value statement towards prospective customers, e.g., in the 'knowledge worker' business, the higher the economic benefit generated by consistent presentations.

The economic benefit of using a DSML does not only depend on presentations that share similar subjects. Instead, it also allows for taking advantage of similarity on a higher level of abstraction: The more subjects can be represented with the same modelling language, the higher the reach of reuse of a corresponding DSML. In addition to that, the use of a DSML will be the more valuable, the higher the demand for graphical representations that are consistent with respect to the represented content and to the graphical notation. Again, this will be especially the case in settings, where integrity and coherence of particular graphical representations are regarded as part of the value expected by the targeted audience of presentations. Furthermore, the analytical power offered by DSML is the more valuable, the more gainful corresponding analyses are – especially those that are supported by tools. This can be expected to be the case whenever problems are targeted that are characterised by a high degree of complexity which can be effectively reduced by models created through a DSML. Note that there are scenarios that hardly justify the use of a PDMS, e.g., the quick creation of drafts that serve one particular occasion only.

For prospective users the effort to get productive with a PDMS is remarkable, which is especially

the case for using a DSML. Therefore, PDMS will be appreciated by prospective users only, if their use offers convincing incentives. The more often a user creates presentations that are similar to existing ones, the more likely it is that he acknowledges the benefits of reusing existing work, which in turn also depends on the size and quality of the available repository. Furthermore a user's appreciation of a more sophisticated tool will trend to increase with his level of professional education. To exploit the potential enabled by a PDMS, there is also need for a supportive organisational context. The more an organisation – or its specific culture – puts emphasis on quality and coherence of knowledge and its presentation, the more likely it should support the introduction and use of a PDMS. In addition to that, the acceptance of a PDMS will also be fostered by an organisational culture that appreciates sharing of knowledge and the use of methods.

8 Conclusion and Future Work

The work presented in this paper did not result from a dedicated research project. Instead, it was rather a side product from our research on enterprise modelling. While it promises to enrich business presentations and make them clearly more than today a source and medium for organisational knowledge management, its realisation faces remarkable challenges. Only partially, these are related to the realisation of a PDMS itself: a (pragmatic) conception of model variants and a more sophisticated repository management are demanding, but feasible. The DSML that currently exist are related to enterprise modelling. Further topics and DSML accounting more specifically for markets and products are required. To get a systematic overview of topics that may be suited for a PDMS, studies of existing business presentations, e.g., within large consultancy firms, should be helpful. The presented tool environment served to demonstrate a prototypical approach to implementation that makes use of an existing framework. Other approaches are feasible as well. For instance, 'in place editing' as it is featured by various compound document architectures could be used to include diagrams in a presentation. Whenever a diagram would require editing, a corresponding editor would be launched.

While modifying and enhancing the set of DSML is supported by the metamodelling environment, the current solution remains unsatisfactory because generating code from a metamodel jeopardises the existing code base. The respective challenge is targeted by various approaches. Some are aimed at representing conceptual models in code, e.g., Balz and Goedicke (2009), Wada and Suzuki (2005). However, they are limited by the semantics of the corresponding programming languages. For our purpose, approaches that are aimed at programming languages that provide a distinct meta layer are more promising (Bettin and Clark 2010; Ducasse and Gîrba 2006). The meta layer allows for adapting a progrmaming language to a corresponding modelling language. Especially the approach in Bettin and Clark (2010) might be an interesting foundation for revising the tool because it can be integrated with Eclipse. We are currently investigating this option.

The main challenge, however, is not related to technology or research: Only if the evolution of the knowledge base – both within one organisation and in a market of specialised providers – succeeds, the full benefit of PDMS, including economies of scale, can be accomplished. One approach to this challenge could be the establishment of 'open presentation' initiatives. Various platforms for sharing presentation files indicate that many professionals are motivated to participate in such initiatives. Furthermore, the success of the proposed solution will chiefly depend on user acceptance. As we outlined above, acceptance can be expected to depend on various factors. To develop a more thorough understanding of these factors, empirical studies with prospective users and corresponding organisations are required. However, they need to account for the specific epistemological challenges that relate

to investigating linguistic preferences mentioned above already.

Independent from the domain of business presentations, there is a further domain that could benefit from smart slides. A plethora of slides is produced for teaching purposes. Using DSML for this purpose would not only foster reuse and productivity, it would also provide IS students with an interactive learning environment in a twofold sense. On the one hand, it would serve to present learning material. On the other hand, the underlying conceptual foundation could serve as a showcase for system design.

References

Andersson B., Bergholtz M., Edirisuriya A., Ilayperuma T., Johannesson P., Gordijn J., Schmitt M., Abels S., Hahn A., Wangler B., Weig H. (2006) Towards a Reference Ontology for Business Models. In: Proceedings of the 25th International Conference on Conceptual Modeling (ER), pp. 482–496

Andersson B., Johannesson P., Zdravkovic J. (2009) Aligning goals and services through goal and business modelling. In: Information Systems and E-Business Management (ISeB) 7(2), pp. 143–169

Balz M., Goedicke M. (2009) Embedding Process Models in Object-Oriented Program Code. In: Proceedings of the 1st Workshop on Behaviour Modelling in Model-Driven Architecture (BM-MDA). Enschede

Bettin J., Clark T. (2010) Advanced Modelling Made Simple with the Gmodel Metalanguage. In: Proceedings of the First International Workshop on Model-Driven Interoperability (MDI). Oslo

Ducasse S., Gîrba T. (2006) Using Smalltalk as a Reflective Executable Meta-language. In: Nierstrasz O., Whittle J., Harel D., Reggio D (eds.) Proceedings of Model Driven Engineering Languages and Systems, 9th International Conference, MoDELS. LNCS Vol. 4199. Springer, Genova, Italy, pp. 604–618

Eckerson W. (2005) Performance Dashboards: Measuring, Monitoring, and Managing Your Business. Wiley & Sons, Hoboken, NJ

Few S. (2006) Information Dashboard Design: The Effective Visual Communication of Data. O'Reilly, Beijing

Frank U. (2002) Multi-Perspective Enterprise Modeling (MEMO) – Conceptual Framework and Modeling Languages. In: Proceedings of the Hawaii International Conference on System Sciences (HICSS-35). (digital edition). Honolulu

Frank U. (2006a) Evaluation of Reference Models. In: Fettke P., Loos P. (eds.) Reference Modeling for Business Systems Analysis. Idea Group, Hershey, pp. 118–140

Frank U. (2006b) Towards a Pluralistic Conception of Research Methods in Information Systems Research. ICB Research Report 7. Institute for Computer Science and Business Information Systems (ICB), University of Duisburg-Essen. Essen

Frank U. (2011) The MEMO Meta Modelling Language (MML) and Language Architecture. 2nd Edition. ICB Research Report 43. Institute for Computer Science and Business Information Systems (ICB), University of Duisburg-Essen. Essen

Frank U., Lange C. (2007) E-MEMO: A Method to support the Development of customized Electronic Com-merce Systems. In: Information Systems and E-Business Management (ISeB) 5(2), pp. 93–116

Frank U., Strecker S. (2009) Beyond ERP Systems: An Outline of Self-Referential Enterprise Systems. ICB Research Report 31. Institute for Computer Science and Business Information Systems (ICB), University of Duisburg-Essen. Essen

Frank U., Heise D., Kattenstroth H. (2009) Use of a Domain Specific Modeling Language for Realizing Versatile Dashboards. In: Tolvanen J.-P., Rossi M., Gray J., Sprinkle J. (eds.) Proceedings of the 9th OOPSLA workshop on domain-specific modeling (DSM). Helsinki Business School, Helsinki http://www.

dsmforum.org/events/DSM09/Papers/Frank.pdf

Gordijn J. (2004) E-business value modelling using the e3-value ontology. In: Curry W. L. (ed.) Value creation form e-business models. Elsevier Butterworth-Heinemann, Oxford, UK, chap. 5, pp. 98–127

Hevner A. R., March S. T., Park J., Ram S. (2004) Design Science in Information Systems Research. In: MIS Quarterly 28(1), pp. 75–105

Jung J. (2007) Entwurf einer Sprache für die Modellierung von Ressourcen im Kontext der Geschäftsprozessmodellierung. Dissertation. Logos, Berlin

Kirchner L. (2008) Eine Methode zur Unterstützung des IT-Managements im Rahmen der Unternehmensmodellierung. Logos, Berlin

Laforcade P. (2007) Graphical representation of abstract learning scenarios: the UML4LD experimentation. In: Spector J. M., Sampson D. G., Okamoto T., Kinshuk, Cerri S. A., Ueno M., Kashihara A. (eds.) Proceedings of IEEE International Conference on Advanced Learning Technologies (ICALT 2007). IEEE Computer Society, Niigata, Japan, pp. 477–479

Maier R. (2004) Knowledge Management Systems. Information and Communication Technologies for Knowledge Management, 2nd ed. Springer, Berlin, Heidelberg, New York

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

Osterwalder A. (2004) The Business Model Ontology. A Proposition in a Design Science Approach. PhD thesis, University of Lausanne

Paquette G., Léonard M., Lundgren-Cayrol K., Stefan Mihaila S., Gareau D. (2006) Learning Design based on Graphical Knowledge-Modeling. In: Educational Technolog & Society 9(1), pp. 97–112

Sampson D., Karampiperis P., P. Z. (2006) Authoring Web-Based Learning Scenarios Based on the IMS Learning Design: Preliminary Evaluation of the Ask Learning Designer Toolkit. In: International Conference on Computer Systems and Applications (AICCSA). IEEE Computer Society, Dubai, pp. 1003–1010

Schauer H. (2008) Unternehmensmodellierung für das Wissensmanagement. Eine multiperspektivische Methode zur ganzheitlichen Analyse und Planung. Dissertation, University Duisburg-Essen

Tantau T. (2007) User Guide to the Beamer Class, Version 3.07 http://latex-beamer.sourceforge.net

Tufte E. (2006) The Cognitive Style of PowerPoint: Picking Out Corrupts Within, 2nd ed. Graphics Press: Chesire/CT

Wada H., Suzuki J. (2005) Modeling Turnpike Frontend System: A Model-Driven Development Framework Leveraging UML Metamodeling and Attribute-Oriented Programming. In: Briand L., Williams C. (eds.) Proceedings of Model Driven Engineering Languages and Systems, 8th International Conference, MoDELS. Springer, Montego Bay, pp. 584–600

Ulrich Frank

Research Group Information Systems and Enterprise Modelling
University Duisburg-Essen
Germany
ulrich.frank@uni-due.de