Artur Caetano, António Rito Silva, José Tribolet

Business Process Decomposition

An Approach Based on the Principle of Separation of Concerns

The functional decomposition of a business process breaks it down into progressively less granular activities. Decomposition contributes to the modular design of a system, the reuse of its parts and to its overall comprehensibility. But achieving these qualities requires a business process to be decomposed consistently, which implies it is always split into an identical set of activities according to a specific purpose, regardless of the modeller's and modelling context. This paper describes an application of the principle of role-based separation of concerns to consistently decompose a business process into its constituent atomic activities, thus separating its distinct features and minimising behaviour overlap. An activity is abstracted as a collaboration between role types that are played by entities. The decomposition method successively separates the overlapping roles until an activity is specified as a collaboration of an orthogonal set of role types. The method facilitates the consistent decomposition of a business process and the identification of its atomic activities. The relevance of the method is assessed through a number of scenarios according to the guidelines of design science research.

1 Introduction

It is widely accepted that one of the fundamental problems in the design and development of knowledge-based systems is extracting information from the experts and then translating it to the form of some knowledge base in order to attain a given purpose. As in the case of business process modelling, this transformation is not straightforward as the source knowledge is often not structured or formalised and tends to be of complex nature. Furthermore, the purpose of the model itself may not be well defined or understood by all of its stakeholders. As a matter of fact, a number of researchers posit that complexity is an essential property of design activities in general due, in part, to the inevitably incomplete formulation of the problem and to our inability to cope simultaneously with all of the constraints of a given problem (Dasgupta 1991).

Business process models translate the knowledge about how an organisation operates. These models are a fundamental piece of enterprise architecture as they support the communication, analysis, implementation and execution of the organisation's structure, business, systems and technology (Lankhorst 2006; Op't Land et al. 2009). Process models also provide the means to analyse alternative designs intended to align the business with the information systems and technology.

However, the task of process modelling must cope with the multiple views and goals of the different organisational stakeholders while capturing the complex relationships between a number of elements, such as information, people, goals and systems, as well as the underlying control and data flows. These models are often produced by merging the partial contributions of different teams which are involved in the elicitation and analysis of the organisation's processes. These teams probably have varying levels of experience and may even employ different modelling techniques. Other factor is that the name given to the activities and entities of a process tend to be the primary means to communicate and understand their semantics. However, the naming procedure is usually ad hoc since similar activities may be

described with different terminology because the verbs and nouns used to describe a process are seldom derived from a shared ontology. Put together, these factors tend to lead to models that lack consistency.

A business process model is deemed consistent if the same modelling principles explicitly apply to all parts of the model. Inconsistent process models display several problems, such as (1) heterogeneous schemes for naming its activities and entities (2) usage of different modelling styles and (3) process hierarchies with arbitrary depth and level of detail. Inconsistent models are not only hard for their users and stakeholders to understand but also hamper the tasks of process analysis, redesign, reuse and automation as they may lead to erroneous interpretations of the process content and may ignore relevant information.

Decomposition deals with breaking down a system into progressively smaller subsystems that are responsible for some part of the problem domain (Dietz 2006; Weber 1978). The specification of a business process encompasses a set of activities that are structured according to control or data dependencies. Each activity is a transformation function that maps inputs to outputs. This approach abstracts an activity as a black-box as it describes its external behaviour and produces models that conceptually divide a system into a hierarchy of functions. Thus, the functional decomposition of a process entails its recursive separation into a set of more detailed activities. The lowest level of decomposition consists of indivisible atomic activities.

Consistent business process decomposition can significantly improve the comprehensibility and minimise the omission of relevant information in a model (Huber et al. 1990). Decomposition is also a means to modularise large systems and to facilitate the reuse of partial models as it reduces coupling, favours the compactness of the specification and allows for multiple levels of detail to co-exist (Bass et al. 1998). As a consequence, models become easier to understand and communicate, which, in turn, facilitates their validation, redesign and optimisation (Odell 1998).

The depth of decomposition of a process depends on the purpose and scope of the model. Each decomposition level describes process elements from a different abstraction level. Higher-level process models formulate an overview of its activities and are usually used as means to facilitate its analysis and communication. Lower-level models providing detailed descriptions that can be used to identify supporting systems and application services. In either case, users have to maintain particular modelling requirements, such as homogenous abstraction of process element names on the same decomposition level, in order to keep the model consistent. This implies identifying an appropriate point to stop the decomposition as a means to avoid extraneous detail. Consequently, maintaining this particular modelling requirement demands a significant amount of experience in the field of process engineering and may result in an extra analysis effort.

To address these issues, this paper presents a general method that specifies how to decompose a business process according to the concerns that are involved in the specification of its activities. A business process activity is constructed as a collaboration of roles played by entities. Each role abstracts the behaviour that an entity displays in the context of an activity. The set of roles played by an activity defines its potential decompositions and the functional views on that process.

The remainder of this paper is structured as follows. Section 2 introduces the basic concepts of natural type, role type and activity. Sections 3 and 4 describe the functional decomposition method and the underlying role ontology along with a running example. Section 5 reviews related work and section 6 summarises the research methodology behind this project. Finally, section 7 concludes the paper with a summary of our approach and an outlook on future work.

2 Fundamental Concepts

Role modelling is a separation of concerns technique that is used in multiple areas of knowledge such as data modelling (Bachman 1980), object-oriented and conceptual modelling (Kristensen 1995; Reenskaug et al. 1996; Steimann 2000), framework design (Riehle 2000), business process modelling (Krogstie et al. 1997; Ould 1995) and enterprise architecture (**L\^e+'06**; Caetano et al. 2009; Uschold et al. 2000; Wegmann 2003; Zacarias et al. 2009; Zacarias et al. 2007).

With role-based business process modelling an activity (a business verb) is abstracted as a set of collaborations between entities (business nouns). The entities represent the things that are of interest in a specific modelling context. Each unit of behaviour of an entity is abstracted as a role and, as a result, activities are defined as a collaboration pattern between roles. If no roles are being played within the system then there are no collaborations and, therefore, no activities to be modelled.

Figure 1 shows the relationships and cardinalities between four entities involved in the assemble product process which we will use as a running example to illustrate the concepts outlined above. The activity assemble product is defined by the collaboration pattern between the roles being played by the entities part, assembling machine, product and person.

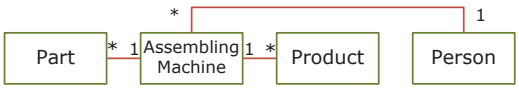

Figure 1: Relationships between entities

The activity describes how a product is assembled from a number of parts by means of an assembling machine. The activity is semi-automated as the machine is operated by a person.

As shown in Figure 2, the relationships between the entities result in one collaboration context where a natural type displays a specific behaviour

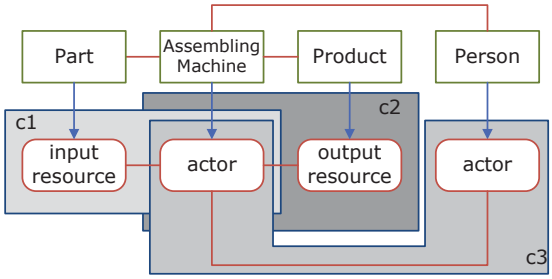

Figure 2: Roles and contexts of collaboration

(Caetano et al. 2009; Zacarias et al. 2009). Such behaviour is abstracted as a role type. Thus, in the first collaboration context (c1) each part plays the role of input resource in their relationship with the assembling machine which, in its turn, is playing the actor role. In context c2, the assembling machine produces the assembled product, i.e., the product is the output resource of this actor. Finally, in context c3, the person relates to the machine as its actor. The collaboration between these four roles uniquely defines the assemble product activity as depicted in Figure 3. The actor role states that an entity is able to perform some action in the context of an activity. The resource role states that an entity which is playing it can be used or consumed (input resource) or created (output resource) during the performance of an activity. The remainder of this section details the concept of entity (natural type), role (role type) and activity.

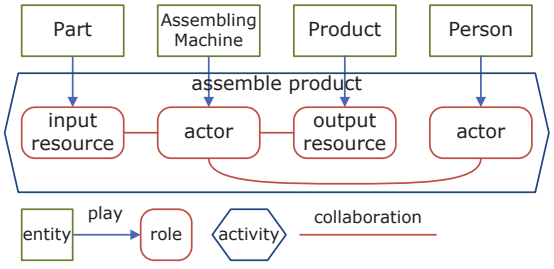

Figure 3: Role-based specification of the activity assemble product

2.1 Natural Types and Role Types

Sowa (1984) distinguished between natural types 'that relate to the essence of the entities' and role types 'that depend on an accidental relationship to some other entity'. By developing Sowa's ideas further, Guarino et al. (1994) presented an ontological distinction between these two types. This distinction is based on the concepts of foundedness and semantic rigidness. A type is considered founded if its specification implies a dependency or relation to some other individual. A type is semantically rigid if the identity of an individual depends on the type assigned to it. If the type is removed from the individual then it cannot be further identified nor classified. Thus, a type is not semantically rigid if it can be assigned to and removed from an individual without changing its identity. Based on these definitions, a type that is both founded and not semantically rigid is a role type. In contrast, a natural type is characterised by being semantically rigid and not founded.

To illustrate the above classification properties, let us take the example of Figure 3 and classify the concepts of person and actor as either natural or role types. First, let us focus on the 'foundedness' of these concepts. Actor is a founded type since for something or someone to be assigned the actor type there must be something being acted upon. Conversely, the person type is not founded since it exists on its own terms. It defines the identity of the individual to which it is assigned to, regardless of its relationships with any other individual. Thus, the person type is not founded whereas the actor type is founded.

Regarding 'semantic rigidness', the actor type is not semantically rigid because its identity is independent of the individual to whom the type is assigned to. This means the actor type is not able to identify the individual by itself. On the other hand, the person type is semantically rigid as its own identity is directly coupled to the individual's identity. Therefore, actor is a role type (founded and not semantically rigid) whereas person is a natural type (not founded and semantically rigid).

2.1.1 Natural Types

Entities are natural types. In enterprise modelling, an entity describes all things an organisation deems relevant to store some information about for a specific purpose and in the context of a specific model. These include concepts such as persons, places, machines and products. According to the definition of natural type, an entity can always be unambiguously identified and defined in isolation, i.e., without any relationship with other types. Entities can be classified according to its intrinsic features. Entities may relate structurally to other entities, as in the case of an entity which is composed by other entities (e.g., an order is composed of items).

2.1.2 Role Types

A role type, or role for short, is the observable behaviour of an entity in the scope of a specific collaboration. Different roles separate the different concerns that arise from the collaborations between entities. Hence, a role represents the external visible features of that entity when it collaborates with another entity in the context of an activity. An entity relates to other roles through the play relationship. An entity that plays no roles is not participating in any activity since it is not able to produce actual behaviour. An entity enters the role when it starts playing it and leaves the role when the specific behaviour specified by the role is concluded. Each role adds a set of external features to an entity in the context of that collaboration.

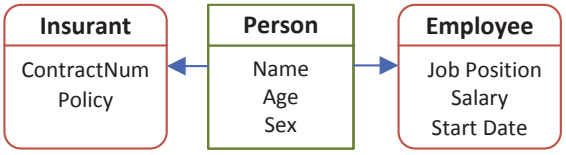

Figure 4: Feature augmentation

Consider the case depicted in Figure 4 where the entity Person plays two roles: Employee and Insurant. Person has a number of intrinsic features such as name, age and sex that are part of its natural type. But while the Person plays the Employee role feature space is augmented with a set of specific features, such as the job position and salary. Another specific set of features (such as Contract Number and Policy) are transiently added to the Person entity while it plays the Insurant role in a different context. These distinct sets of external features depend on the specific role being played by the entity. This approach effectively separates the entity's feature space since its intrinsic features are differentiated from each of the external features that transiently relate to an entity through the roles it plays.

2.2 Activities

A business process is an ordered execution of activities that produces goods or provides services that add value to the organisation's environment or to the organisation itself. Thus, modelling a business process involves specifying the set of activities that define its operation and the flow that defines how the activities are coordinated.

An activity is specified by a collaboration of role types. It is a behaviour element that describes part of the functionality available to the organisation. Since a role type separates the description of the intrinsic features of an entity from the features that derive from the collaborations it participates in, the specification of an activity itself is independent of the specification of the entities playing the roles.

Figure 3 depicts the assemble product activity as a unit of functionality that result from the collaboration between a set of roles. However, this activity model is conceptual as it may have been specified from a different perspective or with a different level of detail, which would have implied using a different role ontology. The granularity level of the activities is also arbitrary as it is always possible to add more detail to its specification. Hence, the naming of an activity is actually irrelevant for the purpose of its specification as the role collaboration pattern is the only means to specify it unambiguously. Therefore, an activity is uniquely identified by the collaboration of roles that are involved in its specification. Two activities are deemed equivalent if and only if they share the same set of role collaborations.

3 Functional Decomposition

The decomposition of a business process results in a set of functional sub-activities, each of which can be recursively decomposed. Thus, the functional behaviour of a whole process can be constructed upwards from the lowest level of decomposition towards the top-level activity. The lowest level of decomposition describes primitive or atomic activities that cannot be further divided. Actually, the related literature describes different approaches to the functional decomposition of processes but does not provide the means to unambiguously identify atomic activities nor the functional decomposition mechanisms that provide consistent decomposition results (cf. section 5).

The approach proposed in this paper is to use role types as the criteria for process decomposition. This means each decomposition step separates a different concern (i.e., a role type) from the other concerns that specify the activity. An activity is deemed atomic, meaning it cannot be further decomposed, when all of its concerns are effectively separated. This translates to having no overlapping role types in the activity's specification. It also implies that the classification of an activity as atomic actually depends on the role ontology that is being utilised to generate the process model. So, different role ontologies yield different decomposition criteria and, thus, different process models.

The algorithm decompose (`S, R`) recursively separates an activity into sub-activities as long as there are overlapping concerns. `S` is the ordered set of all the roles' type instances used in activity

Table 1: Pseudo-code for the decomposition algorithm

```
decompose (S, R)
    D ← ∅
    decompose' (S, R, D, 0)
    decompose ← D
end
decompose' (S, R, D, level)
    if R ≠ ∅ then
        R0 ← firstElementOf(R)
        Dlevel ← ∅
        if countInstancesOfType(R0, S) > 1 then
            for all r ∈ R0 do
            Sd ← (S – R0) ∪ r
            Dlevel ← Dlevel ∪ Sd
            decompose' (Sd, R – R0, D, level +1)
            end for
        else
            decompose' (S, R – R0, D, level +1)
        end if
        D ← D ∪ { Dlevel }
    end if
end
```

to be decomposed. The set `R` (which is a subset of the types of `S`) contains the role types that define the domain to be used to decompose the activity. If all the role types in `S` are included in `R` then all roles will be separated. The role types not included in `R` will remain overlapped after the decomposition. The output of decompose (`S, R`) is a set of sets. Each of these sets represents an activity, with the outer set representing the first level of decomposition. As described in the pseudo-code shown in Table 1, the algorithm works recursively on the set `S` according to the roles defined in `R`. The symbol level identifies the current decomposition level with 0 representing the top level activity. The symbol `D` represents the output set of the decomposition and D_{level} is the set of decomposed activities pertaining to a given level of depth. The algorithm makes use of two additional functions not detailed here: `firstElementOf(X)` returns the first element of the set `X`; `countInstancesOfType(t, X)` counts the number of instances of the type `t` within the set `X`.

To illustrate activity decomposition, Figure 5 depicts activity `A1` according to four collaborations `a, b, c,` and `d,` occurring between role types `R1, R2 and R3`. Each collaboration is an instance of these role types. Activity `A1` is specified by:

```
S = { a:R1, b:R1, c:R2, d:R3, e:R3 }
```

The domain `R` used to decompose the activity is defined as:

```
R = { R1, R2, R3 }
```

Since all the types in `R` are included in `S`, the activity will be fully decomposed, i.e., no overlapping role types will remain.

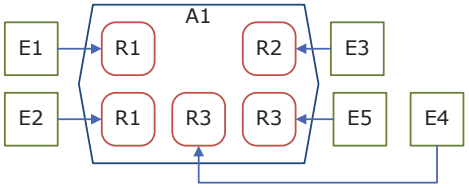

Figure 5: Activity A1 according to roles R1, R2, R3

Applying the algorithm above to decompose `A1` according to `R`, i.e., `decompose(S, R)`, results in

```
D = { D1, D2 }
```

`D1` is the first level of decomposition of `A1` and separates the `R1` role type instances. Thus, `D1` comprises two activities:

```
D1 = {(a:R1, c:R2, d:R3, e:R3),
      (b:R1, c:R2, d:R3, e:R3) }
```

`D2` it is the second level of decomposition of `A1`. It further decomposes the activities in `D1` that still display overlapping concerns. After decomposition, `D2` comprises four activities:

```
D2 = {(a:R1, c:R2, d:R3),
      (a:R1, c:R2, e:R3),
      (b:R1, c:R2, d:R3),
      (b:R1, c:R2, e:R3)}
```

The activities in D2 display full separation of concerns since no role types are overlapped. Therefore, the four activities in D2 are atomic, meaning they cannot be further divided according to the role type domain defined by R.

Assume now the role ontology defined by R1, R2, R3 describes locations, goals and actors. This means R1 stands for the Locator role, a geographical location, R2 is the Goal role, that models the intended state of the affairs to be achieved after executing the activity, and that R3 is the Actor role, which describes the action of someone or something performing the activity.

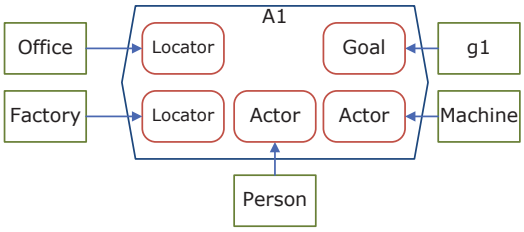

Figure 6: Activity A1 with Actor, Locator and Goal roles

Decomposing A1 according to the Locator role (R1) yields the set D1 with two activities, A1.1 and A1.2, as shown in Figure 7.

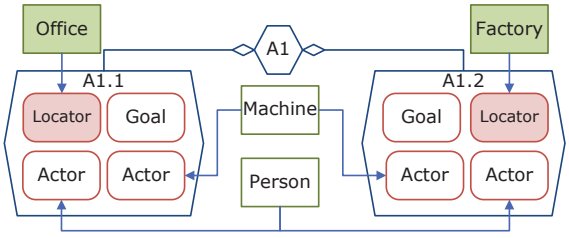

Figure 7: Decomposition of A1 over the Locator role

Each of these activities separate A1 according to the location concern. Decomposing A1 according to the Actor role (R3) would also produce two activities, each focusing on the specific operations of each actor involved in A1. Note that A1 cannot be decomposed according to the Goal role (R2) as this concern does not overlap with any other role of the same type. The concerns in activities A1.1 and A1.2 can be further separated as shown in Figure 8 and Figure 9.

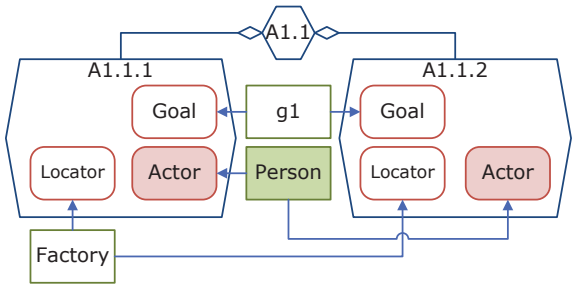

Figure 8: Decomposition of A1.1 over the Actor role

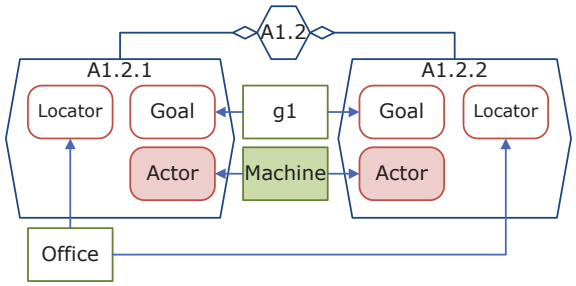

Figure 9: Decomposition of A1.2 over the Actor role

The decomposition of A1 according to the role type domain R =(Locator, Actor, Goal) results in the set D2 with four atomic activities, each focusing on a specific concern:

```
A1.1.1=(Office:Locator,
        Person:Actor, g1:Goal)
A1.1.2=(Factory:Locator,
        Person:Actor, g1:Goal)
A1.2.1=(Office:Locator,
        Machine:Actor, g1:Goal)
A1.2.2=(Office:Locator,
        Machine:Actor, g1:Goal)
```

Note that A1 cannot be further decomposed according to these three roles. Further decomposition of the activity A1 is only possible if additional overlapping concerns are included in its specification.

This approach to activity decomposition is consistent as each level of decomposition can be reproduced according to a set of explicit criteria. As a consequence, a business process can always be systematically separated into its constituent

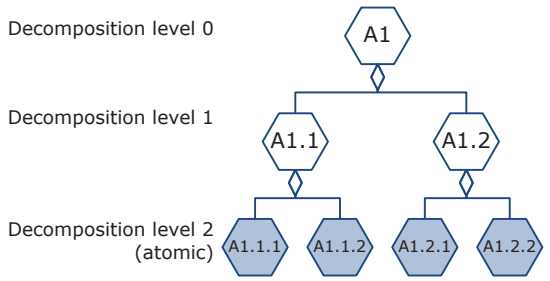

Figure 10: Decomposition of activity A1.

atomic activities. Additionally, the condition for activity decomposition is also explicit, as the procedure stops whenever all of its overlapping role types are separated.

4 Role Ontology

The decomposition method introduced in the previous section relies on using a role type ontology to model the collaborations taking place in the context of a business process. The ontology defines the set of role types required to model a specific domain and the relationships between these types.

A business process can be modelled from different perspectives according to the model's goals and purpose. Although there are multiple classification schemes that allow categorising the modelling perspectives, we posit that these always crosscut the six orthogonal linguistic interrogatives (how, what, where, who, when, why) that are used as columns in the Zachman framework (Zachman 1987). These interrogatives can be used to create a number of perspectives, including the following (Carlsen 1996; Giaglis 2001):

- **Functional** represents what activities are being performed in the context of a given process.
- **Informational** represents what entities (i.e., data or resources) are being manipulated by the activities of a process.
- **Behavioural** represents when activities are performed and how they are performed, usually through the specification of the process orchestration.
- **Organisational** represents why an activity is being performed (i.e., it represents the goals it achieves), where it is performed and by whom.

It is important to emphasise that these perspectives and role types are just given as an example of what can be defined as a role ontology. The actual perspectives and role types must be designed according to the purpose of the model, the requirements of the stakeholders and the domain of the organisation.

4.1 Role Types

The basic role types intend to describe the six modelling interrogatives as summarised in the next Table and sections.

Table 2: Basic concerns and corresponding roles

Concern	Role type	Section
who	actor	4.1.1
what	resource	4.1.2
where	locator	4.1.3
why	goal, rule	4.1.4
how	starter, finisher	4.1.5, 4.2
when		

4.1.1 'Who' Concern

An actor role represents the action of an entity that does some task in the context of an activity. Actors are played by entities which represent people, computer systems, mechanical tools or any other devices that produce active change within an organisation. A possible specialisation scheme of the actor role type focuses on its nature, such as: social actor (people or organisations), application actor (computational or non-computational applications that are used to perform a task) and hardware actor (computer hardware, machines and other devices that support the application and social actors). Another specialisation scheme, which is orthogonal to the actor's nature, includes roles such as operator, auditor and supervisor.

Using the actor role as the criterion for activity decomposition results in the ensuing sub-activities separating who is responsible for its operation since each sub-activity focuses on describing the actions of an individual actor. For instance, the decomposition of the assemble product activity (cf. Figure 3) according to the actor role would create two activities, one for the actions being performed by the person and another for the actions of the machine.

4.1.2 'What' Concern

A resource is the role played by an entity when manipulated by an actor in the context of an activity. A resource specialisation scheme that focus on how a resource is transformed within an activity consists of two roles: input resource role and output resource role. The former can be further specialised as consumed resource role and used resource role, whereas the latter can be specialised as created resource role and refined resource role. Once again, other orthogonal schemes are possible, such as classifying a resource according to its existence (e.g., tangible, intangible, information, physical, etc.).

Using the resource role as the criterion for activity decomposition, results in the ensuing sub-activities describing how each individual resource is being transformed. The decomposition of the activity described in Figure 3 according to this concern separates the two overlapping input resource roles.

4.1.3 'Where' Concern

The locator role captures the geographical or the logical location of an entity. The sub-activities of an activity that is decomposed according to the locator role are operated in different locations.

4.1.4 'Why' Concern

A goal represents a measurable state of affairs that the organisation intends to achieve. The entity that specifies such state of affairs plays the goal specifier role, which relates to the goal fulfiller role. Goals are usually achieved by the entities playing the actor or resource role. A rule asserts conditions or operating parameters that an activity must comply with. The entity that specifies the constraint plays the rule specifier role which relates to the rule complier role.

4.1.5 'How' and 'When' Concerns

The behavioural perspective can be captured by the starter and finisher roles. The first models the event that triggers the start of an activity while the second signals its completion. These two roles can be used to describe how the activities of a process are orchestrated, as described in the next section.

4.2 Activity Orchestration

Orchestration is a concern that captures how and when the activities within a process are sequenced. This is modelled by the specification of the constraints that limit how activities are ordered. These constraints include control flow constructs, such as the AND-split, XOR-split, and the AND-join, which are widely covered by business process and workflow modelling languages (Aalst et al. 2003; Russell et al. 2006).

The ontology used to capture the activity orchestration concern makes use of the role types finisher and starter. The finisher role is played whenever the activity completes, i.e., when the collaboration between its roles comes to an end. Conversely, the starter role signals the start of the activity. Thus, these two roles capture the successor-predecessor relationships occurring during the process orchestration. Each control flow construct is therefore modelled as an activity. This section describes the modelling of two constructs, sequence and AND-split, but the same reasoning here described would also apply to modelling other control flow constructs. A sequence or sequential flow defines a basic successor-predecessor constraint, i.e., an activity starts after the completion of another activity.

The AND-split creates multiple successors by dividing the flow into multiple branches. The following activities can then be executed concurrently and be synchronised together later using an AND-join or a similar synchronisation construct.

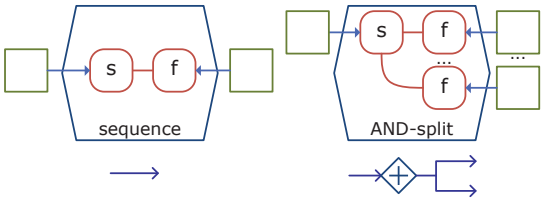

Figure 11: Sequence and AND-split

Figure 11 shows the role collaborations within the sequence and AND-split activities. Sequence is an activity defined by the collaboration between the finisher and starter roles. The starter role is played by an entity whenever the previous activity terminates its execution. This entity is usually a resource that has been generated by the activity or an event that signals its termination. As the sequence construct is unconditional, the starter role immediately follows the finisher role. The AND-split is modelled similarly, but it is the collaboration between a starter role with at least two finisher roles.

As a result, each control flow construct is modelled as an activity, which, in its turn, is modelled as a collaboration between starter and finisher role types. This means the same role-based principles are used for modelling the activities within a business process as well as the activities that specify the control flow.

Figure 12 illustrates this approach. The top part represents the business process specification that includes four activities A1–A4, each modelled through the collaboration between entities playing roles. The bottom part of the figure represents the control flow specification, which includes two activities: sequence and AND-split. The middle part of the figure depicts how activities A1–A4 are orchestrated using the sequence and AND-split constructs.

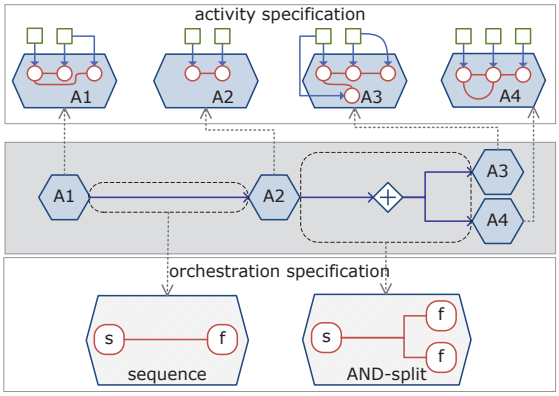

Figure 12: Activity orchestration

5 Related Work

Functional decomposition is supported at language level by most process modelling languages, including ArchiMate (Lankhorst 2006), BPMN (OMG 2008), EPC (Scheer 2000), and IDEF-0 and IDEF-3 (Mayer et al. 1995). The decomposition of subsystems through the hierarchic classification of process models has also been applied to Petri nets (Reisig and Rozenberg 1998) and BPEL (Kloppmann et al. 2005). Although these approaches make possible creating a hierarchical representation of a process, their intent is not the definition of techniques for consistent activity decomposition but, instead, the representation of generic decomposition structures. Nevertheless, the shortcomings of the lack of consistency in process decomposition and in the identification of its atomic activities are pointed out by several authors (Davis and Brabdänder 2007; Ingvaldsen and Gulla 2006).

Some top-down approaches make use of reference models to describe how a process is structured into a hierarchy of activities. For instance, the Supply-Chain Operations Reference-model (SCOR) describes three levels of detail to assist the definition and configuration of an organisation's supply chain (Bolstorff and Rosenbaum 2008). The Process Clarification Framework defines a hierarchical decomposition of business processes which is 3–4 levels deep and crosses 12 operating and management categories (APQC

2008). Other approaches, such as the ARIS framework (Scheer 2000), describe processes as value-added chains or as chains of events and tasks and prescribe the levels of detail for decomposition. The first two decomposition levels address the business viewpoint of the model, the next 3–4 levels focus on the structure of process operation and the lower level describes the procedural details of the tasks.

A different set of approaches relies on algorithmic methods to analyse the process specification and to determine its consistency. One of these methods introduces similarity measures that are derived from the syntactic and structural features of the process (represented with Petri nets) in order to detect inconsistencies between its activities (Ehrig et al. 2007; Hornung et al. 2008). The measures make use of a linguistic ontology to evaluate the similarity between the names of the activities thus assisting the detection of decomposition anomalies. Another approach defines how to perform the structured functional decomposition of activities that are specified with AND / OR trees or production rules (Soshnikov and Dubovik 2004). Process mining techniques extract information from existing event logs and enable the discovery of business processes (Aalst et al. 2005). These bottom-up mining techniques make possible verifying the conformance of a model derived from an event log against an existing model as well as identifying the atomic activities of a process (Aalst et al. 2007).

Other approaches that make use of ontologies to specify business processes (e.g., Greco et al. 2004; Uschold et al. 2000) also lack the means to uniquely identify atomic activities and to consistently decompose a process.

Altogether, and to the best of our knowledge, existing approaches do not define the necessary means to consistently decompose a business process and to unambiguously identify the atomic activities that constitute the process.

6 Research Methodology

The methodology behind the results reported in this paper is grounded on design science (Hevner et al. 2004; March and Smith 1995). Design science focuses on the development of solutions for practical problems. This contrasts with the development and the verification of theories as in behavioural science methodologies.

Research on enterprise architecture, modelling and engineering fits the design science paradigm as its focal goal is not building information systems but creating methods and techniques to analyse, model, and understand the horizontal and vertical interfaces between the business, systems and technology (Braun et al. 2005). The essential tangible result of a design science project consists in creating an artefact that addresses a particular issue that is relevant to a certain group of stakeholders. In this context, Hevner et al. (2004) proposed a set of guidelines to conducting design science projects. The following points briefly summarise how these were applied to this work.

- **Design as an artefact**. This project deals with applying the principle of separation of concerns to business process modelling. This paper describes an artefact that deals with business process decomposition role modelling as a separation of concerns mechanism.
- **Problem relevance**. The artefact enables the consistent decomposition of a business process. By doing so, it addresses several problems that are relevant in enterprise engineering in general and business process modelling in particular. We emphasise the following problems: (1) how to systematically identify the atomic activities of a process; (2) how to make explicit the principles behind process decomposition; (3) how to make decomposition dependent on the specification of the process and not on the modelling team experience.
- **Design evaluation**. This paper makes use of a small set of scenarios Hevner et al. (2004) built around the artefact to demonstrate its utility.

- **Research contributions**. The paper describes an algorithm for business process decomposition that enables the consistent decomposition of business processes. The paper also describes the applicability of a role-based separation of concerns technique to business process modelling.
- **Research rigour**. The artefact was designed to address a problem identified in the enterprise engineering and business process modelling literature. The solution is grounded on the principles of role modelling, separation of concerns, and business process modelling.
- **Communication of research**. The research is reported through publications aimed at the practitioners and researchers within the enterprise engineering area and mainly targets business process modellers.

7 Conclusion and Future Work

Activity decomposition is an abstraction technique that enables the modularisation of business processes. A decomposed process is more intuitive and easier to understand as each decomposition step incrementally focuses on a smaller number of overlapping concerns. This fosters reusing the process models and also increases the ability to communicate and analyse each model. However, each decomposition step must provide a consistent level of detail so that the resulting set of atomic activities comprising the lowest level of decomposition is always the same regardless of the modelling team's experience.

The aim of our approach is to guide the procedure of process decomposition so that decompositions are explicit and consistent. The proposed method supports the decomposition of business processes according to the separation of overlapping concerns. Business processes are modelled as the collaboration of natural types (entities) that play roles in the context of activities. Each concern is described as a role type. A role ontology sets the role type domain and constrains the possible decomposition space. This approach enables activities to be consistently decomposed. It also makes possible to uniquely identify and discriminate the atomic activities of a process according to a specific role ontology.

The future work related to this research project includes the specification of the role ontologies that translate a number of specific constructs derived from mainstream business process modelling languages. We are also currently developing a set of case studies that aim to demonstrate the advantages and drawbacks of our method implementation when used to model and decompose large-scale business processes.

References

APQC (2008) APQC Process Clarification Framework (PCF)- Consumer Products, version 5.0.2, 10/04/2008. http://www.apqc.org/pcf.

Aalst W. v. d., Beer H., Dongen B. v. (2005) Process Mining and Verification of Properties: An Approach based on Temporal Logic. In: Meersman R. (ed.) OTM Confederated International Conferences. Springer, Berlin, Germany, pp. 130–147

Aalst W. v. d., Reijers H., Weijters A., Dongen B. v., Medeiros A. A. d., Song M., Verbeek H. (2007) Business Process Mining: An Industrial Application. In: Information Systems Journal 32(5), pp. 713–732

Aalst W. v. d., Hofstede A. H. M. t., Kiepuszewski B., Barros A. (2003) Workflow Patterns. In: Distributed and Parallel Databases 14(1), pp. 5–51

Bachman C. (1980) The role data model approach to data structures. In: Deen S. M., Hammersly P. (eds.) Proceedings of the International Conference on Data Bases. Heyden & Son, pp. 1–18

Bass L., Clements P., Kazman R. (1998) Software Architecture in Practice. Addison-Wesley, Reading, Massachusetts

Bolstorff P., Rosenbaum R. (2008) Supply Chain Excellence: A Handbook for Dramatic Improvement Using the SCOR Model 2nd ed.

Springer, Berlin

Braun C., Wortmann F., Hafner M., Winter R. (2005) Method construction – a core approach to organizational engineering. In: ACM Symposium on Applied Computing. Santa Fe, New Mexico, USA, pp. 1295–1299

Caetano A., Rito Silva A., Tribolet J. (2009) A Role-Based Enterprise Architecture Framework. In: 24th Annual ACM Symposium on Applied Computing, ACM SAC 2009. Hawaii, USA

Carlsen S. (1996) Comprehensible Business Process Models for Process Improvement and Process Support. In: Constantopoulos P., Mylopoulos J., Vassiliou Y. (eds.) Advances on Information Systems Engineering, 8th International Conference, CAISE'96. vol. 1080 LNCS Springer, Crete, Greece

Dasgupta S. (1991) The Nature of Design Problems. Cambridge University Press, Cambridge, UK, pp. 13–35

Davis R., Brabdänder E. (2007) ARIS Design Platform: Getting Started with BPM. Springer, London, UK

Dietz J. L. G. (2006) Enterprise Ontology: Theory and Methodology. Springer, Berlin

Ehrig M., Koschmider A., Oberweis A. (2007) Measuring Similarity between Semantic Business Process Models. In: Roddick J. F., Hinze A. (eds.) Conceptual Modelling 2007, Proceedings of the Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007). Australian Computer Science Communications, Victoria, Australia, pp. 71–80

Giaglis G. M. (2001) A Taxonomy of Business Process Modeling and Information Systems Modeling Techniques. In: International Journal of Flexible Manufacturing Systems 13(2), pp. 209–228

Greco G., Guzzo A., Pontieri L., Sacca D. (2004) An Ontology-Driven Process Modeling Framework. In: Galindo F., Takizawa M., Traunmuller R. (eds.) 15th International Conference on Database and Expert Systems Applications. IEEE Computer Society, Zaragoza, Spain, pp. 13–23

Guarino N., Carrara M., Giaretta P. (1994) An Ontology of Meta-Level Categories. In: Kaufmann M. (ed.) Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94). San Mateo, CA, pp. 270–280

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

Hornung T., Koschmider A., Lausen G. (2008) Recommendation Based Process Modeling Support: Method and User Experience. In: Li Q., Spaccapietra S., Yu E., Olivé A. (eds.) 27th International Conference on Conceptual Modeling (ER'08). Springer, Barcelona, pp. 265–278

Huber P., Jensen K., Shapiro R. M. (1990) Hierarchies in Coloured Petri Nets. In: 0th International Conference on Application and Theory of Petri Nets. vol. 483 LNCS Springer, pp. 313–341

Ingvaldsen J. E., Gulla J. (2006) Model Based Business Process Mining. In: Journal of Information Systems Management 23(1) Special Issue on Business Intelligence, pp. 19–31

Kloppmann M., Koenig D., Leymann F., Pfau G., Rickayzen A., Riegen C. v., Schmidt P., Trickovic I. (2005) WS-BPEL Extension for Subprocesses BPEL-SPE

Kristensen B. (1995) Object-Oriented Modeling with Roles. In: 2nd International Conference on Object-Oriented Information Systems., pp. 57–71

Krogstie J., Carlsen S., Consulting A., Chicago I. L. (1997) An integrated modelling approach for process support. In: 30th Hawaii International Conference on System Sciences, HICSS 1997., pp. 189–198

Lankhorst M. (2006) Enterprise Architecture at Work: Modelling, Communication and Analysis. Springer, Berlin

March S., Smith G. (1995) Design and natural science research on information technology. In: Decision Support Systems 15, pp. 251–266

Mayer R., Menzel C., Painter M., deWitte P., Blinn T., Perakath B. (1995) Information Inte-

gration for Concurrent Engineering - IDEF3 Process Description Capture Method Report Interim Technical Report April 1992-September 1995, Knowledge Based Systems Inc. Knowledge Based Systems Inc

OMG (2008) Business Process Modeling Notation (BPMN) Specification. v 1.1 (formal/2008-01-17). January 2008

Odell J. (1998) Advanced Object-Oriented Analysis and Design Using UML. Cambridge University Press, Cambridge

Op't Land M., Proper E., Waage M., Cloo J., Steghuis C. (2009) Enterprise Architecture: Creating Value by Informed Governance The Enterprise Engineering Series. Springer, Heidelberg

Ould M (1995) Business Processes: Modeling and analysis for re-engineering and improvement. John Wiley & Sons, Chichester

Reenskaug T., Wold P., Lehn O (1996) Working With Objects: The OOram Software Engineering Method. Manning Publication Co., Greenwhich

Reisig W., Rozenberg G. (1998) Lectures on Petri Nets: Basic Models, vol. 1491 LNCS. Springer, Heidelberg

Riehle D. (2000) Framework Design: A Role Modeling Approach. PhD thesis, Swiss Federal Institute of Technology, Zurich, Switzerland

Russell N., Hofstede A. t., Aalst W. v. d., Mulya N. (2006) Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22

Scheer A. W. (2000) Business Process Modeling 3rd ed. Springer, Berlin

Soshnikov D., Dubovik S. (2004) Structured Functional Decomposition Approach to Knowledge-Based Business Process Modeling. In: 6th Joint Conference in Knowledge-Based Software Engineering (JCKBSE 2004). IOS Press, Protvino, Russia

Sowa J. (1984) Conceptual Structures: Information Processing in Mind and Machine. Addison-Wesley, New York

Steimann F. (2000) On the representation of roles in object-oriented and conceptual modelling. In: Data & Knowledge Engineering 35(1), pp. 83–106

Uschold M., King M., Moralee S., Zorgios Y. (2000) The Enterprise Ontology. In: The Knowledge Engineering Review 13(01), pp. 31–89

Weber H. (1978) Modularity in Data Base System Design: A Software Engineering View of Data Base Systems. In: VLDB Surveys, pp. 65–91

Wegmann A. (2003) On the systemic enterprise architecture methodology. In: International Conference on Enterprise Information Systems (ICEIS 2003). Angers, France

Zacarias M., Magalhães R., Pinto H., Tribolet J. (2009) An agent-centric and 'context-aware' perspective for the alignment between individuals and organizations. In: Information Systems Shasha D., Vossen G. (eds.)

Zacarias M., Caetano A., Magalhães R., Pinto H. S., Tribolet J. (2007) Towards Organizational Self-Awareness: An Initial Architecture and Ontology. In: Rittgen P. (ed.) Ontologies for Business Interactions. Idea Group Inc

Zachman J. (1987) A Framework for Information Systems Architecture. In: IBM Systems Journal 26(3), pp. 276–292

Artur Caetano, António Rito Silva, José Tribolet

Center for Organizational Design and Engineering
INESC Inovação and Department of Computer Science and Engineering
Instituto Superior Técnico, Technical University of Lisbon
Avenida Rovisco Pais. 1
1049-001 Lisboa
Portugal
{artur.caetano | rito.silva | jose.tribolet}@ist.utl.pt