An Application of Role Modelling to the Decomposition of Business Processes

Artur Caetano[1,2,3], José Borbinha[1,2], José Tribolet[1,3]

[1] Instituto Superior Técnico, Technical University of Lisbon,
Avenida Rovisco Pais 1, 1049-001 Lisboa, Portugal.
[2] Information Systems Group, INESC-ID, Lisboa, Portugal.
[3] Centre for Organizational Design and Engineering, INESC INOV, Lisboa, Portugal.
artur.caetano@ist.utl.pt, jose.borbinha@ist.utl.pt, jose.tribolet@ist.utl.pt

Abstract. A business process can be decomposed into a hierarchy of finer-grained activities through the application of functional decomposition. This approach contributes to the modularity and reusability of a process and its parts. However, mainstream business process modelling languages do not prescribe methods to systematically decompose a process. As a result, the procedure and the criteria used to functionally decompose a process are usually undefined or implicit, thus limiting the traceability and the validation of the design. This paper proposes a method to functionally decompose a process using the principles of role modelling. Role modelling is a separation of concerns strategy that abstracts a system according to distinct features with minimum overlap. The objective of the proposed method is to enable the identification of the atomic activities of a business process and the specification of business process hierarchies. This paper describes the decomposition method, the role-based ontology used as decomposition criteria and an example of application.

Keywords. Business process modelling, role modelling, ontology, separation of concerns, enterprise engineering.

1 Introduction

This paper presents an application of separation of concerns to the decomposition of business processes. This approach enables the consistent design of business process hierarchies and the consistent identification of the atomic activities of a business process. Separation of concerns is a means to tackle problems in multifaceted domains. It addresses the different constraints of the problem domain individually (i.e. separates the concerns) and then attempts to combine the partial solutions as a single solution to the problem. But for this approach to yield satisfactory results the concerns that are being separated must be fairly independent to start with so that they do not overlap with one another and thus facilitate the separation process. Furthermore, the problem solving activity itself needs to yield partial solutions that are composable as an overall solution. As such, this abstraction strategy attempt to reduce complexity by factoring out details so that one can focus on a few concepts at a time [1]. Systems can be abstracted according to the principles classification, generalization and decomposition.

In particular, decomposition deals with breaking down a system into progressively smaller subsystems that address specific parts of the problem domain.

A business process specification defines how its activities are organized as a process that adds some kind of value to an internal or external customer. The activities of a process may be designed using different approaches, such as conversation-based or transformation-based [2]. Transformation-based design is used in mainstream business process modelling languages such as BPMN and event-driven process chains. It focuses on resource transformation and on the control and data flow structures behind such transformation. In term of process specification it describes the flow of activities that use input resources to generate output resources (v. Figure 1).

Each business process activity can be decomposed as a set of finer-grained activities, thus creating a layered specification that hides complexity. This abstraction strategy conceptualizes the external behaviour of an activity as a functional black-box where the internal (white-box) construction details are hidden. Therefore, it produces models that conceptually divide a system into a hierarchy of functions. This approach grounds many business process modelling languages (e.g. [3], [4], [5]).

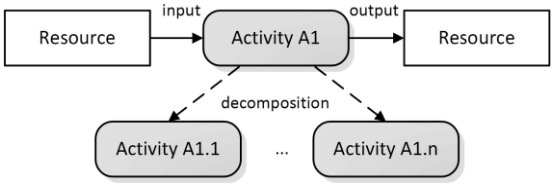

Figure 1. A transformational input–process–output model.

Business process models capture the knowledge about how an organization operates. These models are a fundamental piece of its enterprise architecture since they support the communication, analysis, implementation and the execution of the organization's operations. Process models also provide the means to analyse alternative designs intended to align the business with the information systems and supporting technology. However, the task of process modelling must cope with the multiple views and goals of the different organizational stakeholders while capturing the complex relationships between disparate elements, such as information, people, goals and systems. These models are often produced by merging the partial contributions of different teams, with different backgrounds and experience, who are involved in the elicitation, modelling and analysis of an organization's processes. Put together, these factors add to models that lack consistency. Examples of inconsistency include using arbitrary criteria for functionally decomposing a process, depicting different levels of detail, using different modelling styles, and using incoherent criteria for identifying and naming the activities and the information pertaining to a process. Inconsistent process models are not only hard for their users to understand but also hamper the tasks of process analysis and redesign as they may leave out relevant information or lead to erroneous or ambiguous interpretations of the process content.

Consistent business process decomposition can improve the clarity and the overall model integrity as well as minimizing the omission of relevant information [6]. Decomposition is also a means to modularize large systems through a reduction of module coupling and an increase of module cohesion thus facilitating module reuse. It also favours the compactness of a specification as it allows multiple levels of detail to

co-exist [7]. As a consequence, models become easier to understand and communicate and that simplifies their validation, design and optimization.

The level of the functional decomposition of a process depends on the modeller and on the purpose and scope of the process model. Each decomposition level describes process elements from a different abstraction level. To do so, users have to satisfy specific modelling non-functional requirements, such as homogenous abstraction of process element names on the same decomposition level in order to improve model consistency.

This paper presents a method that specifies how to decompose a business process according to the concerns that are involved on 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 the process' decomposition space and thus the superset of functional views that can be generated.

This paper follows a design science research approach, focussing on the development of effective solutions for practical problems. The tangible results of a design science project consist in artefacts that address a particular issue that is relevant to a group of stakeholders [8]. The research questions behind this project are the following:

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 experience of the process modellers?

The remainder of this paper is structured as follows. Section 0 reviews related work. Section 3 introduces role modelling and its core concepts. Section 4 describes the application of role modelling to the functional decomposition of business processes and section 5 describes how a role ontology can be used to guide decomposition. Section 6 shows an example of application and section 0 concludes the paper.

2 Related Work

Functional decomposition is supported by most process modelling languages such as BPMN [3], EPC [9] and IDEF-0/3 [10] as well as process modelling languages that make use of object-oriented principles such as ArchiMate [11]. The decomposition of subsystems through the hierarchic classification of process models has also been applied to Petri nets [12] and BPEL [13]. Although these approaches make possible to create a hierarchical representation of a process, their intent is not the definition of specific methods for consistent activity decomposition but the representation of generic decomposition structures. This means that these languages do not address the issue of the consistency of the hierarchical structure as they tend to be method-independent. Yet, the shortcomings of the lack of consistency in process decomposition have been pointed out by some authors [14, 15].

To partially address this problem, reference models can be used to guide the task of process decomposition, thus increasing the consistency of the resulting process hier-

archies. For instance, the Supply-Chain Operations Reference (SCOR) model describes three levels of detail to assist the definition and configuration of an organization's supply chain [16] [17]. The Process Clarification Framework (PCF) defines a hierarchical decomposition of business processes that is 3-4 levels deep and crosses 12 operating and management categories [18]. Other approaches, such as the ARIS framework [9], describe processes 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 levels on the procedural details of its tasks. However, the content that is to be included on the specification of each level of decomposition is not defined. Once again, these approaches make possible creating inconsistent decomposition structures and not to specify at all a method that states how these hierarchical structures are to be produced.

An alternative avenue of research makes use of algorithmic approaches to analyse process models and to assess their consistency. An approach relies on using similarity measures derived from the syntactic and structural features of a process represented with Petri nets to detect inconsistencies between the activity specification and to assist the detection of decomposition anomalies [19]. The measures make use of a linguistic ontology to evaluate the similarity between the activity labels. Other approaches use production rules to perform the functional decomposition of activities [20]. Process mining techniques extract information from existing event logs and facilitate the discovery of business processes [21]. These bottom-up mining techniques support the verification of the conformance of a model derived from an event log against an existing model as well as identifying the atomic tasks of a process [22]. However, process mining is primarily based on the analysis of logs and therefore assumes instances of processes are already being supported by the organization's information systems. Other approaches make use of enterprise ontologies to specify business processes but do not focus on the specification of process decomposition [23-25].

Altogether, the approaches that were reviewed do not fully define the means to consistently decompose a process into activities or to unambiguously identify the atomic activities that are part of it. The method described in this paper aims contributing to the problem of design consistent business processes using role modelling as an approach. The next section describes the core concepts behind role modelling and how it relates to process modelling.

3 Role Modelling

This section presents a motivational example that introduces the concepts behind role modelling. A business process is represented as a set of *activities* (business verbs) that abstract collaborations between *entities* (business nouns). The entities represent the things within the organization that are of interest in a specific modelling context. Each unit of behaviour that an entity plays is abstracted as a role [26, 27]. Figure 2 uses BPMN to describe a view over a partial business process showing the data flow of entities `E1`, `E2` and `E3` through the activities `Transform` and `Control Quality` who are respectively performed by actors `Operator` and `Supervisor`.

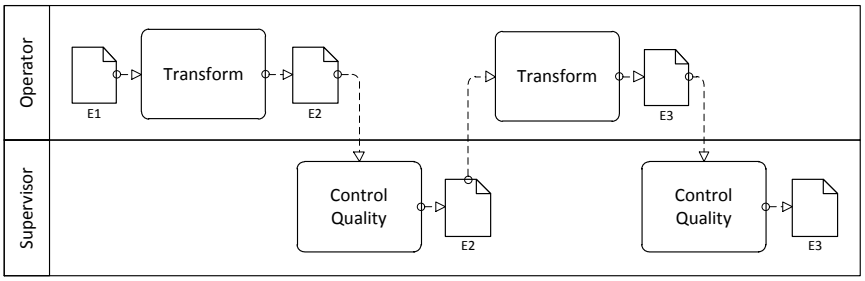

Figure 2. Partial process model represented in BPMN.

These activities can be abstracted as a collaboration of entities that play specific *roles*. If we use the collaborations defined on the classic input-process-output process pattern where an activity transforms inputs into outputs and is performed by an actor playing some role, the behaviour of the activities `Transform` and `Control Quality` may be abstracted or "roleifed" according to a role ontology with roles `Input Resource`, `Output Resource`, `Supervisor` and `Operator`, where the latter two roles are a kind of actor role. In the case of the `Transform` activity this translates to saying that the activity is performed by the `Operator` actor role, its input `E1` is an entity playing the `Input Resource` role and the output `E2` is an entity playing the `Output Resource` role. This is depicted in Figure 3.

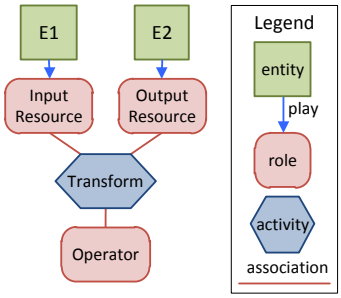

Figure 3. Role-based specification of the Transform activity.

The next diagram shows the result of applying this four role type ontology to the process depicted in Figure 2. Although the resulting role-based model is more complex in terms of number of concepts and relationships than the original BPMN model, its goal is to detail how entities collaborate and to create the foundation to enable the consistent specification of views over the process.

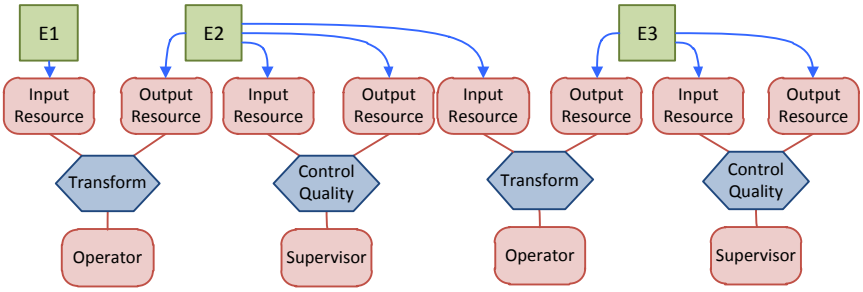

Figure 4. Role-based model of the process described in Figure 2.

In this proposal, all interactions between entities are always mediated by roles which abstract their specific collaborative behaviour. The decomposition depicted above is actually the full decomposition of the process according to the four roles of the ontology: `Input Resource`, `Output Resource`, `Supervisor` and `Operator`. As a result, each of the decomposed activities separates each of these four concerns from all the other concerns.

If the same process is decomposed exclusively according to the "who" dimension through the `Supervisor` and `Operator` actor roles, then a different decomposition will be generated as depicted in Figure 5. This identifies the two activities that separate the behaviour of the two actors within this process and keeps the `Input Resource` and `Output Resource` roles overlapped.

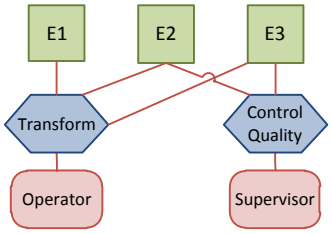

Figure 5. Decomposition according to the Actor roles.

If the process is decomposed according to the "what" dimension through the Input and `Output Resource` roles the decomposition identifies two activities associated with the these roles as depicted in Figure 6.

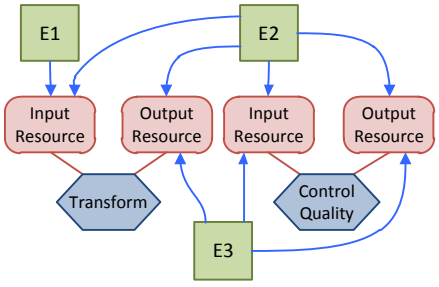

Figure 6. Decomposition according to the Resource roles.

The next section defines the three concepts introduced along with this example, namely role, entity and activity.

3.1 Core Constructs: Roles, Entities and Activities

Sowa [28] distinguished between natural types "that relate to the essence of entities" and role types "that depend on an accidental relationship to some other entity". By developing Sowa's ideas further, Guarino et al. [29] presented an ontological distinction between these two types: a role type is founded and not semantically rigid whereas a natural type is not founded and semantically rigid. Role types require to always stand in some relation to other individual (i.e. the type is founded) and they can enter and leave the type without losing their identity (i.e. the type is not semantically rigid). In contrast, a natural type is characterized by being semantically rigid and not founded.

Let us consider the concept of `actor` and the concept of `person`. An `actor` is an entity capable of performing behaviour. Examples of `actors` are persons, departments or business units. An `actor` is a founded type since for something or someone to enter the `actor` type there must be something being acted upon or operated; i.e., an `actor` must always stands in relation to some specific subject (the subject that is being acted upon). Conversely, the `person` type is not founded since it exists on its own terms regardless of being in a relationship with any other type. Thus, `person` is a not founded type and `actor` is a founded type.

Regarding the type's "semantic rigidness", the `actor` type is not semantically rigid because its identity is independent of entering or leaving the type, i.e. the identity of the `actor` type is external to the type itself. For instance, the concept of student, which is a kind of actor, continues to exist even when someone playing that role leaves the type, i.e. it is no longer a student. In contrast, the `person` type is semantically rigid as its identity is strongly coupled to the type's identity: the identity of a `person` ceases to exist whenever the `person` type is no longer valid. Therefore, `actor` is a role type because it is founded and not semantically rigid whereas `person` is a natural type because it is not founded and semantically rigid.

3.1.1 Natural Types or Entities

An entity is a natural type. An entity describes the business nouns that are considered relevant within the context of an organization for the purpose of satisfying concerns of specific stakeholders. Entities abstract tangible and intangible concepts such as persons, places, machines, products and information. Since entities are natural type they can always be unambiguously identified and defined in isolation, i.e. without any relationship with any other type. Thus, entities can be classified according to its intrinsic properties. Entities may relate structurally to other entities, as in the case where an entity is part of other entity.

3.1.2 Role Types

A role type is the observable behavioural that an entity display within a specific entity collaboration. Different role types separate the different concerns that arise from the collaborations between entities. Hence, a role represents the external properties of an entity when it collaborates with another entity in the context of an activity. An entity relates to zero or more roles through the *play* relationship. An entity that plays zero roles is not participating in any activity, i.e., it is not producing behaviour. An entity enters the role type when it starts playing that role and leaves the role when that specific behaviour is concluded. Therefore, role types are transiently bound to entity types via the play relationship. While an entity is playing a specific role, its properties are augmented by the external properties brought in by that role.

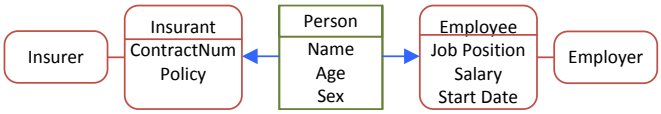

Figure 7. Role types and augmentation of the properties of an entity.

Figure 7 depicts the entity `Person` playing two roles: `Employee` and `Insurant`. In this example, `Person` (an entity type) has three intrinsic properties (`name`, `age`, `sex`) that are part of its natural type. While a `Person` is playing the `Employee` role in collaboration with the `Employer` role its properties are augmented with the `job position` and `salary` properties. These two properties are only relevant and applicable while the `Person` is playing the `Employee` role. Another set of properties (e.g. the insurant's `contract number` and the insurance `policy`) is only brought in while the `Person` plays the `Insurant` role in a different context. Thus, the properties of an entity at a given time instant are the union of its intrinsic properties with the properties brought in by all roles being played by the entity at that instant.

Property augmentation through roles effectively separates the property domain of an entity as its intrinsic properties are seamlessly differentiated from the external properties that transiently relate to entity through the roles it plays.

3.1.3 Activities

A business process is defined as a set of activities that provides goods or services that add value to the organization's environment or to the organization itself. Modelling a business process involves specifying the set of activities that comprise it and the flows that define how the activities collaborate. The same set of activities may collaborate differently in several processes. This means that activity specification and activity collaboration are two independent concerns.

An activity is specified as a collaboration of role types. It is a behaviour element that describes part of the functionality available to the organization. A role type separates the description of the intrinsic properties of an entity from the properties that derive from the collaborations it participates in. Thus, the specification of an activity is not only independent from the orchestration it is involved in but also from the specification of the entities that play the roles pertaining to that activity.

An activity as a unit of functionality that results from the collaboration between a set of roles. This activity's 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 functional specification. Hence, the naming of an activity is actually ancillary 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, meaning two activities are equivalent if and only if they share the same set of role collaborations.

4 Functional Decomposition

The functional decomposition of a business process yields a set of sub-activities, each of which can be further decomposed. The behaviour of a whole process can be constructed upwards through composition from the lowest level of decomposition towards the top-level activity. The lowest level of decomposition specifies the atomic activities of the process that cannot be further divided regarding a given modelling context. Related literature describes different approaches to the functional decomposition of processes but existing approaches do not provide the means to unambiguously identify what makes an atomic activity nor the mechanisms that provide consistent decomposition results.

The approach put forward in this paper is to decompose a process using role types as the main principle for decomposition. Each decomposition step explicitly separates a concern from the other concerns that specify the activity. An activity is deemed atomic, meaning it cannot be further decomposed, when there are no more overlapping roles left in its specification. Therefore, all the similar concerns of an atomic activity are separated. This also implies that the classification of an activity as atomic is contextual as it actually depends on the role ontology that is being utilized to build the model. So, different role type ontologies specify different functional decomposition criteria and, thus, yield a different set of atomic activities. However, the decomposition results are always consistent provided the role types used to drive the decomposed are the same.

The activity decomposition algorithm consists of the function `decompose`. This function recursively adds to its result a set of non-overlapping roles for each level of decomposition. An activity is considered atomic whenever all of its role type instances are non-overlapping. The algorithm is detailed next..

```
decompose(S, R)
begin
   D ← ∅
   decompose'(S, R, D, 1)
   decompose ← D
end
```

```
decompose'(S, R, D, level)
begin
   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
```

Figure 8. Activity decomposition algorithm.

The function `decompose(S, R)` recursively separates an activity into sub-activities as long as overlapping concerns exist. It stops when all atomic activities have been identified according to the decomposition criteria specified in `R`. The set `S` is the ordered set of the roles type instances specified in activity to be decomposed. The set $R \supseteq S$ contains the role types that define the domain to be used as the criteria to decompose the activity. If all role types in `S` are included in `R` then a full role-based separation will ensue and all the atomic activities according to `R` will be identified. The role types not included in `R` will not be separated and remain overlapped after the decomposition. The output of `decompose(S,R)` is a set of sets depicted by the symbol `D` which composed of sets D_{level}. Each D_{level} is the set of decomposed activities pertaining to a given `level` of depth, for instance D_0 is the top level activity, D_1 is the first level of decomposition and so on. The algorithm refers to two unspecified functions: `firstElementOf(X)` returns the first element of the set X and `countInstancesOfType(t, T)` counts the number of instances of the type `t` within the set of types `T`.

5 Role Ontology

The decomposition method relies on the preliminary specification of a role type ontology to describe the collaborations taking place in the context of a business process. The ontology represents 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 to categorize the modelling perspectives, we posit that these crosscut the six linguistic interrogatives 5W+1H, i.e. what, where, who, when, why and how [30]. These interrogatives can be used to create the following modelling perspectives [2, 31]:

1. Functional: represents *what* activities are performed in the context of a process.
2. Informational: represents *what* entities are manipulated by a process.
3. Behavioural: represents *when* activities are performed and *how* they are performed, usually through the specification of the process orchestration.
4. Organizational: represents *why* an activity is performed, *where* it is performed and *by whom*.

The construction of a domain-specific role type ontology entails identifying the concepts and then specifying them as role types. A possible approach is to use the 5W+1H interrogatives as a foundation and specialize these six base role types accordingly. Alternatively, a different set of domain-specific roles can be used as a foundation. In any case, a pragmatic process of creating a role type ontology is incremental and iterative. This process is oriented towards the specific concerns of the stakeholders of a model [32]. This means that decomposing a process into a set of activities must address at least one concern. This also means that every role type included in the ontology is traceable to the concerns that it realizes. Consider a scenario where the concern of decomposition demands identifying the activities performed by different actors. For such task, it suffices defining a single role type that abstracts the concept of `actor`. If this level of detail is deemed insufficient by the model stakeholders then more roles need to be specified. For instance, the `actor` role type may be specialized as the `performer` of a task and as its `supervisor`. Whenever unaddressed concerns arise new role types must be specified and added to the ontology. For instance, consider that besides decomposing a process according to different actors, the decomposition must also separate activities that handle different resources. This would imply adding to the ontology role types such as `resource`. Once again, this role type could be specialized according to the specific requirements of the domain. Possible generic specializations include role types such as `input/output resource`, `tangible/intangible resource`, `information/energy/matter` as well as specialized role types that apply exclusively to a particular organization or vertical business domain. Thus, the ontology can be incrementally constructed by adding role types that realize domain-specific concerns.

As matter of fact, the role types represented next, and which are based on 5W+1H interrogatives, are the result of abstracting a set of domain-independent roles types that were identified throughout a number of modelling projects with different concerns. We have also observed that starting with an empty (or very basic) role type ontology promotes a discussion between the stakeholders of the business process concerning the actual value of adding extra level of detail to the process model. Ideally, the set of role types should be the minimum possible that makes possible generating all the viewpoints required by the stakeholders. In this manner, the architectural concerns are realized by role types, while the overall role type ontology supports the creation of the different model viewpoints.

The next table illustrates a set of roles types that addresses the above concerns according to the six interrogatives.

Table 1. Description of the role ontology.

Role	Concern	Description
Actor	Who	The *actor* role represents the action of an entity that performs a task in the context of an activity. Actors are played by entities such as people, computer systems, mechanical tools or any other devices that produce change within an organization. A specialization scheme of the actor role type focuses on its nature, such as: *social actor* (people or organizations), *application actor* (computational or non-computational applications that are used to perform a task) and *infrastructure actor* (computer hardware, machines and other devices that support the application and social actors). Another specialization scheme, which is orthogonal to the actor's nature, includes roles such as *operator*, *auditor* and *supervisor*.
Resource	What	A *resource* is the role played by an entity when it is manipulated by an *actor* in the context of an activity. A *resource* specialization 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 specialized as *consumed resource* role and *used resource* role, whereas the latter can be specialized as *created resource* role and *refined resource* role. Other orthogonal schemes are possible, such as classifying a resource according to its existence (e.g. tangible, intangible, etc.)
Locator	Where	The *locator* role captures the physical, geographical or logical location of an entity. The sub-activities of an activity that is decomposed according to the locator role are operated in different locations.
Goal	Why	A *goal* represents a measurable state of affairs that the organization intends to achieve. The entity that specifies such state of affairs plays the *goal specifier* role that relates to the goal fulfiller role. *Goals* are usually achieved by the entities playing the *actor* or *resource* role.
Start, Finish	When	The behavioural perspective can be captured through the *starter* and *finisher* roles. The first depicts 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.

6 Example of Application

Figure 9 illustrates an application of the decompose function to activity `A1`. Activity `A1` is defined by the collaboration of role types `R1`, `R2`, `R3`.

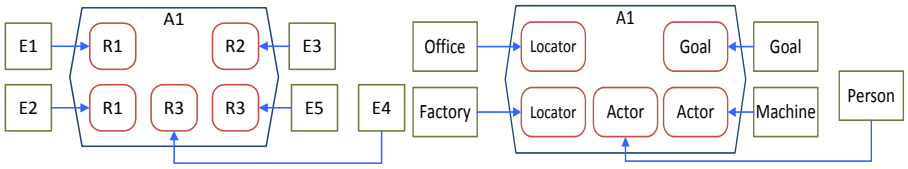

Figure 9. Specification of activity A1 according to roles R1, R2, R3.

Let us consider that role ontology {R1, R2, R3} describes the roles {Locator, Goal, Actor}. Thus, R1 describes a geographical location, R2 models the intended state of the affairs to be achieved after executing the activity, and R3 describes the action of someone of something operating in the context of the activity A1.

Decomposing A1 according to the Locator role R1 yields two activities, A1.1 and A1.2, as shown in Figure 10. Each of these functionally separate A1 according to geographical location concern. Decomposing A1 according to the Actor role R3 produces two activities, each focusing on the specific operations of the actor involved in A1. Note that decomposing A1 according to the Goal role R2 yields no result as this concern does not overlap with any other role. Activities A1.1 and A1.2 can be further separated as shown in Figure 11 and Figure 12.

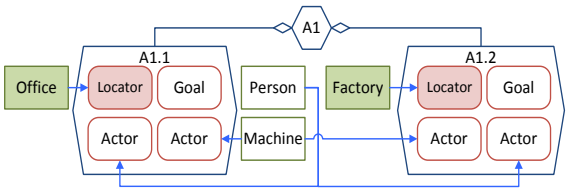

Figure 10. Decomposition of activity A1 on role R1 (Locator).

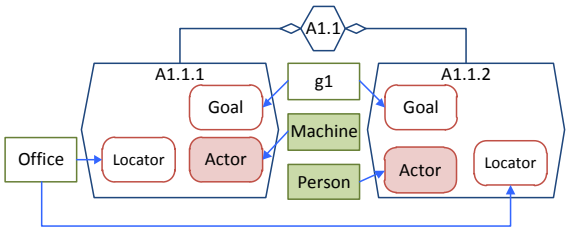

Figure 11. Decomposition of A1.1 on role R3 (Actor).

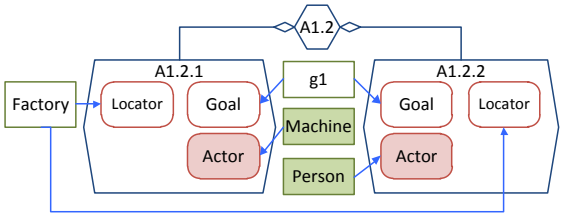

Figure 12. Decomposition of A1.2 on role R3 (Actor).

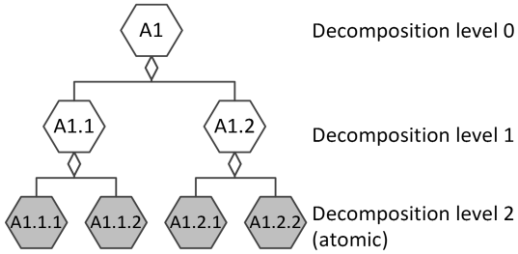

Figure 13. Decomposition tree of activity A1.

Thus, the full decomposition of A1 according to the role tuple (Locator, Actor, Goal) results in four atomic activities (v. Figure 12):

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

Note that activity A1 cannot be further decomposed according to these three roles. Further decomposition is only possible if new role types are added to the ontology or additional overlapping concerns are included in the specification of A1 using the current role types. In terms of the algorithm previously described the above translate to the following steps. A1 is specified by S where:

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

Since the goal is to fully decompose A1, then the set R includes all roles in S, i.e:

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

Applying decompose(S, R) to A1 according to R results in the set D with two levels of depth:

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

D1 is the first level of decomposition and divides activity A1 as:

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

D2 is the lowest level of decomposition and specifies four atomic 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)}
```

In summary, the decomposition mechanism breaks up a process into its constituent atomic activities according to a given set of role types that drive the decomposition. Different decomposition criteria (the criteria is the input parameter R of the decomposition algorithm) yield different decomposition results. In particular, each level of decomposition separates overlapping role type instances. The lowest level of decomposition is such that all role types are fully separated and no further separation is possible given the process specification and the decomposition criteria. As a result of using this approach, a business process can be systematically and consistently separated into its constituent atomic activities according to specific decomposition criteria. Moreover, the motivation behind each decomposition level as well as with each activity pertaining to any given level is explicit as it is fully traceable to the role types used by decomposition algorithm.

7 Conclusions

Activity decomposition is an abstraction technique that enables the modularization of business processes. A decomposed process is easier to understand as each decomposition step incrementally reduces the number of overlapping concerns. This fosters the reuse of the process models and increases the ability to communicate and analyse

them. Each decomposition step must provide a consistent level of detail so that the set of atomic activities comprising the lowest level of decomposition are always coherent, regardless of the stakeholder's requirements and the modelling team's experience.

The aim of this project is to guide the procedure of process decomposition so that decompositions become 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 that play role types in the context of activities. The role ontology specifies the domain of role types and constrains the possible decomposition space. This approach facilitates the consistent decomposition of activities and the identification of the atomic activities of a process according to a given role ontology.

In terms of future work, we are currently working on the specification of role ontologies and the corresponding mechanism to translation to and from mainstream business process modelling languages. We are also currently developing a collaborative software tool to facilitate the management of the role type ontology.

Acknowledgments

This work is partially supported by the EU-FP7 grant 269940, project TIMBUS, "Digital Preservation for Timeless Business Processes and Services".

References

1. Weber, H., Modularity in Data Base System Design: : A Software Engineering View of Data Base Systems. VLDB Surveys, 1978: p. 65-91.
2. Carlsen, S. Comprehensible Business Process Models for Process Improvement and Process Support. In CAISE'96. 1996. Crete, Greece, Springer LNCS.
3. OMG, Business Process Model and Notation (BPMN), version 2.0. 2011.
4. Rosemann, M. and M.P.v.d.A. Wil, A Configurable Reference Modelling Language. Information Systems Journal, 2007. 32: p. 1-23.
5. Brocke, J.v. and M. Rosemann, Handbook on Business Process Management. 2010: Springer.
6. Huber, P., K. Jensen, and R.M. Shapiro. Hierarchies in Coloured Petri Nets. In 10th International Conference on Application and Theory of Petri Nets. 1990. Springer, Lecture Notes in Computer Science, vol. 483.
7. Bass, L., P. Clements, and R. Kazman, Software Architecture in Practice. 1998, Reading, Massachusetts: Addison-Wesley.
8. Hevner, A.R., S.T. March, J. Park, and S. Ram, Design science in information systems research. MIS Quarterly, 2004. 28(1): p. 75-105.
9. Scheer, A.-W., Business Process Modeling. 3rd ed2000, Berlin: Springer Verlag.
10. Mayer, R.J., et al., Information Integration for Concurrent Engineering - IDEF3. April 1992-September 1995, Knowledge Based Systems Inc.
11. The Open Group, ArchiMate 1.0 Specification. ISBN 9789087535025. 2009.
12. Reisig, W. and G. Rozenberg, Lectures on Petri Nets: Basic Models,. 1st ed. Lecture Notes in Computer Science. Vol. 1491, 1998, Heidelberg, Germany: Springer.

13. Kloppmann, M., D. Koenig, F. Leymann, G. Pfau, A. Rickayzen, C.v. Riegen, P. Schmidt, and I. Trickovic, WS-BPEL Extension for Subprocesses BPEL-SPE, 2005, IBM and SAP Joint White Paper.
14. Davis, R. and E. Brabdänder, ARIS Design Platform: Getting Started with BPM. 2007, London, UK: Springer-Verlag.
15. Ingvaldsen, J.E. and J.A. Gulla, Model Based Business Process Mining. Journal of Information Systems Management, 2006. 23(1).
16. Bolstorff, P. and R. Rosenbaum, Supply Chain Excellence: A Handbook for Dramatic Improvement Using the SCOR Model. 2nd 2008, Berlin, Germany: Springer.
17. Supply Chain Council, SCOR 10.0 Model Reference. 2nd 2010: Supply chain Council, Inc.
18. APQC, APQC Process Clarification Framework (PCF)- Consumer Products, version 5.0.2, 10/04/2008, 2008.
19. Hornung, T., A. Koschmider, and G. Lausen, Recommendation Based Process Modeling Support: Method and User Experience, in 27th International Conference on Conceptual Modeling (ER'08), Q. Li, S. Spaccapietra, E. Yu, and A. Olivé, Editors, 2008, Springer: Barcelona, Spain. p. 265-278.
20. Soshnikov, D. and S. Dubovik, Structured Functional Decomposition Approach to Knowledge-Based Business Process Modeling, in JCKBSE 2004, IOS Press: Protvino, Russia.
21. Aalst, W., H. Beer, and B.v. Dongen, Process Mining and Verification of Properties: An Approach based on Temporal Logic, in OTM Confederated International Conferences, R.M.e. al., Editor 2005, Springer: Berlin, Germany. p. 130-147.
22. Aalst, W., H. Reijers, A. Weijters, B.v. Dongen, A.A.d. Medeiros, M. Song, and H. Verbeek, Business Process Mining: An Industrial Application. Information Systems Journal, 2007. 32(5): p. 713-732.
23. Uschold, M., M. King, S. Moralee, and Y. Zorgios, The Enterprise Ontology. The Knowledge Engineering Review, 2000. 13(01): p. 31-89.
24. Greco, G., A. Guzzo, L. Pontieri, and D. Sacca, An ontology-driven process modeling framework,, in 15th International Conference on Database and Expert Systems Applications, F. Galindo, M. Takizawa, and R. Traunmuller, Editors, 2004, IEEE Computer Society: Zaragoza, Spain. p. 13-23.
25. Albani, A., J.L.G. Dietz, and J. Zaha, Identifying Business Components on the basis of an Enterprise Ontology, in Interoperability of Enterprise Software and Applications, D. Konstantas, et al., Editors. 2006, Springer: London. p. 335-347.
26. Caetano, A., A. Rito Silva, and J. Tribolet. A Role-Based Enterprise Architecture Framework. In 24th Annual ACM Symposium on Applied Computing, 2009. Hawaii, USA.
27. Sousa, P., A. Caetano, A. Vasconcelos, C. Pereira, and J. Tribolet, Enterprise architecture modeling with the UML 2.0, in Enterprise Modeling and Computing with UML, P. Rittgen, Editor 2006, Idea Group Inc. p. 67-94.
28. Sowa, J., Conceptual Structures: Information Processing in Mind and Machine. 1984, New York: Addison-Wesley.
29. Guarino, N., M. Carrara, and P. Giaretta, An Ontology of Meta-Level Categories. Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference. Morgan Kaufmann, San Mateo, CA, 1994: p. 270-280.
30. Caetano, A., C. Pereira, and P. Sousa. Generating Multiple Consistent Views from Business Process Models. In LNBIP, Research and Practical Issues of Enterprise Information Systems. 2011. Springer-Verlag, Berlin, Heidelberg.
31. Giaglis, G.M., A Taxonomy of Business Process Modeling and Information Systems Modeling Techniques. Int. Journal of Flexible Manufacturing Systems, 2001. 13(2): p. 209-228.
32. ISO, 42010: Systems and software engineering - Architecture description 2011.