

## Definition and modeling of process using object orientation

Beatriz Terezinha Borsoi and Jorge Luis Risco Becerra

Universidade de São Paulo – Escola Politécnica

Rua Luciano Gualberto, travessa 3, n. 380. CEP 05508-900, São Paulo – SP, Brazil

beatriz.borsoi,jorge.becerra{@poli.usp.br}

### Abstract

Software development processes can be represented as objects because they and the elements that compose them have attributes and operations, can be linked by relationships and have behavior and states. Therefore, this paper proposes a way to represent processes and their elements as objects. This proposal can be used to represent process models benefiting from the object orientation properties and to define tools to automate processes.

### 1. Introduction

Object orientation has been used in many areas, ranging from analysis, project and implementation of software systems [1] to the representation of companies [2]. Software systems are developed by means of processes, and software factories have processes to accomplish their activities related to the software life cycle and to reach their business goals. Thus, object orientation properties are used in this paper to deal with processes as objects and to depict them as object models, such as the models defined using UML [3].

Representing software processes as objects makes it possible to define process models that can be used to standardize processes as objects with identity, attributes and operations, and as a set of objects that exchange information through messages. These process models can be used to define tools to automate process execution and to represent processes and their elements in a structure named architecture, such as object oriented software architecture.

This paper is structured as follows. In Section 2 the concept of process is described. Section 3 presents the proposal of this paper: the object process. Section 4 contains the proposed process essential elements. A case study is presented in Section 5. The paper is concluded with final considerations.

### 2. Concept of process

Software development includes technical and business aspects. Thus, the proposed concept of process is based on the following authors: [4], and [5], who refer to processes in general; [6], [7], [8], [9], [10], and [11], centered on software processes; and [2], aiming to define business processes.

The concepts of process that are proposed by these authors can be summarized as follows: a process is a set of ordered steps to reach a goal, or as a set of activities that produce output from input. A step is an abstraction of an action to be accomplished. The steps can be named sub-processes, activities, tasks, work items or actions. Processes can be hierarchically structured by decomposition of a step in more detailed steps.

Based on the concepts proposed by these authors and the fact

that software development is centered on the accomplishment of activities that compose processes, in this paper a software process is defined as a set of activities structured to reach goals. The activities are carried out by actors that produce and modify artifacts with the help of resources. Activities, roles (as a generalization of actors), resources and artifacts are the elements of the essential conceptualization of the process. These concepts compose the basic semantics to represent processes and they are subject to policies, can assume states and have attributes and behavior.

### 3. Process as object

In the context of this paper, object orientation concepts are applied to the representation of processes because the elements that compose this representation, including the process, can be treated as objects. Two aspects help this understanding: the conceptual perspective, which refers to the concept or the characterization of an object, and the categorization perspective, which refers to the types of objects. The main authors used to this understanding are: [1], [8], [12], [13], and [14].

In spite of the lack of uniformity of these authors in relation to the object concept, in general an object has identity and is composed by attributes and operations. Considering an object as an encapsulation of data and procedures, the object attributes represent the data and the object operations represent the procedures that are applied to the object data.

In relation to the object types, Booch [1] refers to an object as a tangible and/or visible thing that can be intellectually perceived and understood, or something that allows a thought or an action to be directed. Shlaer and Mellor [12] categorize objects in: tangible objects, roles, incidents, interactions, and objects of specification. Rumbaugh et al. [13] classify the objects in concrete and conceptual. To Pressman [8] the objects can be: entities that produce or consume information, things, occurrences or events, roles, organizational units, places, and structures.

According to Booch [1] processes can be considered objects because they have well-defined conceptual boundaries, interact with other processes through operations and exhibit well-defined behavior. Taylor [2] emphasizes the use of object orientation to organize companies and processes. In this perspective, the object remains the main construction block, which represents something with its own characteristics and behavior [15].

In this paper, to consider a process as an object, concepts of object orientation were complemented by publications that refer to the process in the perspective of object orientation or to the process architecture: [2], [16], [17], [18], [19], and [20].

In synthesis, [2] defines a basic class named business element, and from it three classes are derived: organization, process, and

resource; [16] represent a process through product and activity classes; [17] composes a process with artifact, role and activity; [18] define the classes: entity and activity; to [19] there are the classes: role, activity, and work item; and [20] defines the classes: activity, artifact, resource, and agent.

This explanation leads to the understanding that activities named process elements can be defined by means of identity, attributes, and operations, which compose the object as a concept. Therefore, the concept of object and the characteristics of object orientation are applied to the processes and their elements. To Rumbaugh et al. [13] these characteristics are: identity, inheritance, classification, and polymorphism. To Booch [1] they are: abstractions, encapsulation, modularity, and hierarchy.

Identity means that each process and each of its elements are discrete and distinct entities with attributes and behavior. Classification allows elements with the same data structure (attributes) and the same behavior (operations) to be grouped in classes. Through polymorphism the same operation can act in different ways in distinct classes of processes or elements. An operation is an action or transformation that an activity (a process element) performs. Through inheritance, a class that represents processes or elements can be defined as a generic class and to be refined in successive subclasses.

Abstraction consists in disregarding irrelevant aspects to the interest of specific process models or views. Encapsulation allows a software factory to hide processes and activities and to exchange message with other software factories through interfaces. Interfaces define a communication pattern between classes of models. Interfaces describe a set of messages to be followed by the classes that implement these interfaces. Modularity allows processes and activities to be aggregated in autonomous and loosely coupled modules. Hierarchy is a way to organize abstractions and roles.

The concepts of state machine are also applied to processes. These concepts make it possible to apply the sequence of states that a process and its elements are subject to during their lifetime when responding to events. The relationships among process elements can be depicted as UML relationships, such as association, generalization and dependency.

The conceptual conception proposed to the object process is represented in Figure 1, which was defined by: what was described in this section about object orientation in relation to processes, the publications mentioned, and the process conceptual model proposed in [19]. At the core of SPEM [19] is the idea that a software development process is a collaboration between abstract active entities called process roles that perform operations called activities on concrete, tangible entities called work products.

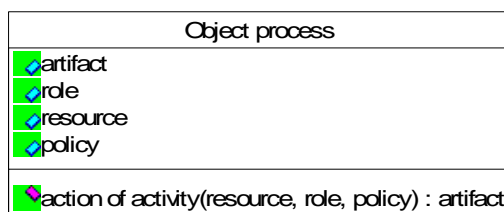


Figure 1. The object process

Figure 1 represents process according to the concepts of object orientation. The process is depicted as an object that has artifact, role, resource and policy as attributes, and action of activity as operation. The action of activity receives resource, role and policy as parameters and returns an artifact. According to the process concept proposed in section 2 this figure contains the main elements of conceptualization and composition of a process. The elementary constructors to represent the object process are defined based on the representation of Figure 1.

Using notation of UML class diagram, Figure 2 depicts the constructors: elements with attributes and operations, behavior and relationship. These constructors supply the semantics to define the notational concepts to compose the models that represent a process as object. In Figure 2, the element is originated from the attributes in Figure 1, the behavior is originated from the operation, which is represented by activity action, and from the relationships among elements.

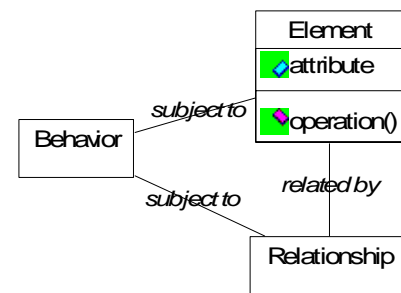


Figure 2. The elementary constructors

The semantics supplied by the representation in Figure 2 shows that the process elements are linked through relationships and have attributes and operations, and that elements and relationships are subject to behavior.

The class element represents the basic block to construct process models. The class behavior refers to the operations accomplished by elements and to the dynamic aspects of the representations. The relationship class has the description of how the elements are related and represents connections of the element with itself, with another element and with many elements forming structures. In this paper, the relationships are defined by UML [3] and by BPMN [21].

The independence between process activities and functions is achieved through process representation as a relationship among objects. The communication based on messages allows the standardization of the interfaces for the communication between processes.

This standardization is important in the integration of software factories. Considering that the communication occurs across software factory boundaries, the distinct competences of the factories are protected against unauthorized access. Therefore, competing software factories can cooperate because interaction occurs by means of messages and interfaces. The integrated activities and processes access only the interface and not the set of properties and actions behind the interface.

3.1 Notational concepts

The notational concepts supply a syntactic and semantic base to represent the metamodels and the models of the processes and their elements according to object orientation, the domain and context, the distribution of processes in the company, the behavior of processes and their elements, and the relationships among processes and elements.

This syntactic and semantic base is complemented by the syntax and the semantics of the language used to depict the metamodels and the models. A language presupposes a vocabulary, which is its basic construction block, and formation rules according to its syntax and semantics. In this paper these blocks are the notational concepts.

The constructors represented in Figure 2 are used to obtain the notational concepts in order to depict the process and its elements as objects. However, there are other representations of the processes using object orientation, although they are not related to the structure or behavior of the process and its elements.

The notational concepts that are related directly with the process and its elements as objects are obtained from the constructors depicted in Figure 2. These concepts can be compared with object notation in UML: processes and their elements are classes; relationships are relationships among classes and objects; behavior is state and interactions between classes and objects; operations are element actions; attributes are characteristics of the elements and of the processes.

The conceptual representation of models, such as domain and context, distribution of processes and role hierarchy, is also defined based on object orientation. However, as they refer to classes and objects without attributes and operations, their models and metamodels are depicted through an abstract generalization of the elementary constructors named concepts.

Thus, the notational concepts are grouped in: concepts and relationships among them, process and its elements as objects, process states, elements states, control and data flows.

Figure 3 presents the notational concepts to depict metamodels and models based on concepts and relationships among them. The syntax to represent these concepts can be, for example, UML class diagram with classes without attributes and operations to represent concepts, and UML relationships to represent relationships.

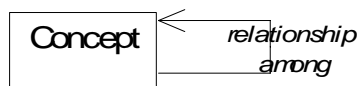


Figure 3. Concepts to represent models

Concepts and the relationships among them are used to represent the domain and context of processes, process distribution in the organizational structure and role hierarchy. The interest of these models is not in the structure and behavior of processes and their elements, but in the conceptualization of processes, process distribution and roles hierarchy.

In order to abstract only the most relevant concepts, it is suggested the use of object orientation properties, such as

classification, generalization and aggregation. However, the use of these properties must be subordinated to the process interests and goals, which effectively determine what must be considered in each model.

The notational concepts to represent static aspects of the process elements are centered in processes and their elements as object. These concepts are derived from the elementary concepts of Figure 2. According to Figure 4, process elements have attributes and operations, and are linked through relationships. UML class diagram and object diagram can be used to depict these concepts: classes and objects with attributes and operations represent processes and their elements; and relationships represent links among elements.

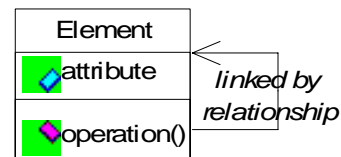


Figure 4. Concepts for the static representation of processes and their elements

The attributes define process and element properties, and represent abstraction of the possible data types of the process and element instances. The operations are abstractions for the execution of the element actions. These actions are services that can be requested by some element. The call of an operation can modify the value of the attributes and the element and/or process state.

The elements and their relationships can be viewed as a process image of one moment or of a period of time. The notational concepts for the dynamic representation of the process elements include process and element state, activity flow and data flow.

Figure 5 presents the notational concepts used to represent state metamodels of processes and their elements. These concepts refer to the behavior depicted in Figure 8. UML state diagrams can be used to represent state models. These diagrams depict the process and the life cycle of its elements, representing their internal dynamics.

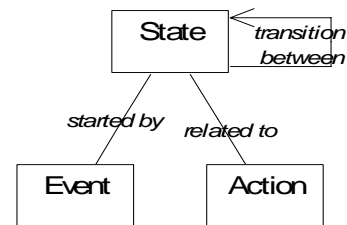


Figure 5. Concepts to represent states of process and its elements

Figure 6 presents the notational concepts to depict the data and control flow metamodel. These concepts refer to the behavior depicted in Figure 8. The models obtained from this metamodel can be depicted in a BPMN or UML activity diagram, for example.

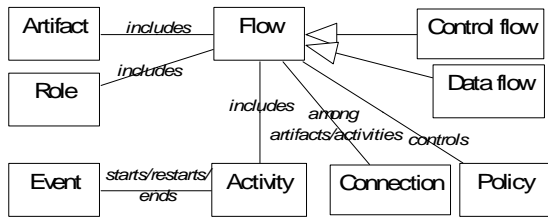


Figure 6. Concepts to represent control and data flows

Policies can represent controls and are used to control the connections and the transition flow among activities. The control flow has a mechanism that allows or disallows the passage of the flow among activities according to certain conditions. The data flow, which represents the connections among artifacts produced and used by the activities, can be subject to the same conditions of the control flow. A connection defines the conditions so that activities are in certain states.

3.2 Essential elements of the process

The essential elements of a process are the elements that answer the questions defined as fundamental to the process and which compose its conceptualization: how it is done, who does it, what is used to do it, what is done, and when it is done. These questions are related to the concept of process proposed in Section 2. The essential elements defined in this paper and their relationships with these fundamental questions are:

- a) "How it is done" is related to the activities that compose processes and are carried out to produce artifacts.
- b) "Who does it" refers to the roles fulfilled by human or automated authors. These authors use resources to carry out activities and to produce artifacts.
- c) "What is used to do it" refers to the resources used by the authors to carry out activities.
- d) "What is done" is related to the artifacts produced and modified by the authors using resources and through activities.
- e) "When it is done" refers to the flow of activities aiming to reach goals.

These questions allow a process to be understood as a set of activities, carried out by authors (roles) that use resources to produce artifacts.

Figure 7 presents the notation of process essential elements using UML class diagram. These elements are obtained from concepts depicted in Figure 4.

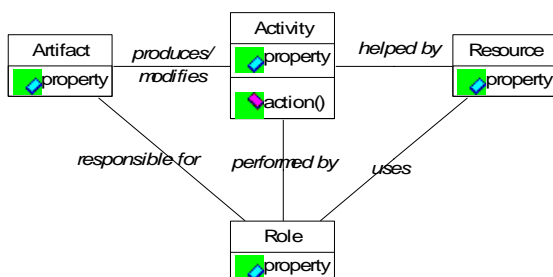


Figure 7. Essential elements of the process

By relating the representations of Figure 7 and Figure 4, it follows that: activity, artifact, role and resource are elements; performed by, responsible for, uses, helped by and produces/modifies are relationships; action is operation; and property is attribute.

According to Figure 7 the elements activity, artifact, role and resource are objects that have attributes and behavior and are linked by relationships. Action represents the operation of the activity element. In this paper decompositions of the activity, such as task, are not defined. However, considering the use of object orientation concepts, an activity is composed when it represents a process or a set of activities; an activity is simple when it is an atomic unit of work.

Aggregated to the essential elements are the complementary elements. The definition of these elements is dependent on the process characteristics, such as its type, its context and domain, and the quality standard used. Figure 8 contains the complementary elements of the process. These elements are obtained from the concepts of Figure 4.

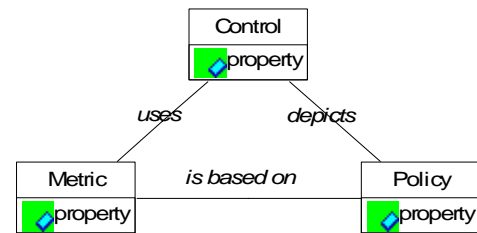


Figure 8. Complementary elements of the process

In this paper, controls, policies and metrics were defined as complementary elements. The controls and policies include quality standard, rules and other criteria that limit the accomplishment of the activities, the action of the roles, the use of resources, and the production of artifacts. The metrics are measures linked to the controls and they aim to verify the adaptation and applicability of the policies.

By relating the representations of Figure 7 and Figure 4, it follows that: control, metric and policy are elements; uses, depicts, is based on are relationships; and property is attribute.

Policies, as a set of rules related to a particular purpose, can be expressed as obligations, permissions, authorizations or prohibitions. The policies prescribe allowed and required behavior, helping in the description of ideal and desirable behavior of the processes and their elements, and in the definition of parameters to the metrics. The controls control the occurrence of behavior.

5. Case study

This section reports an example, through a case study, of the proposed object process, its notational concepts, and complementary and essential elements. This object process was used in the definition of metamodels and models of processes of two software factories (Factory A and Factory B) for the development of the same software project. In this paper metamodels are models used to define models. It is relevant to emphasize that only some metamodels and models produced in

the case study are in this section.

Figure 9 presents the domain and context model. This model contains the basic conceptualization that is used to guide the definition of other models and to provide a common understanding of the context to the integrated software factories.

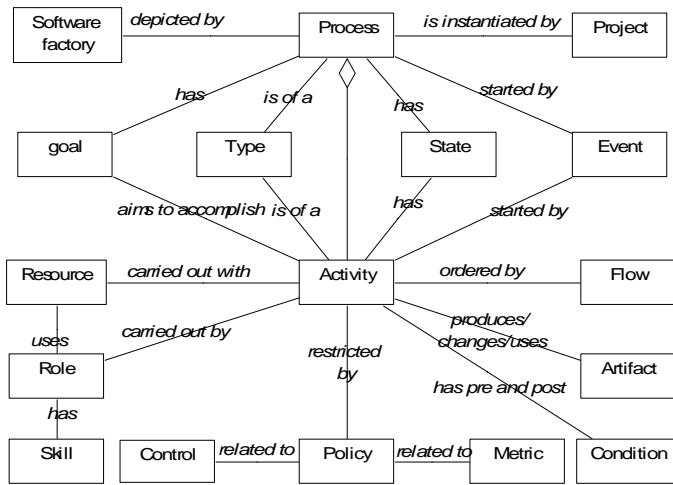


Figure 9. Domain and context standard model

A conceptual domain helps to standardize the understanding of the process models and their elements, and the necessary communication and compatibility to integrate software factories. Figure 9 presents a generic view of the concepts that represent the domain according to the context of two software factories integrated to develop the same software project. This context is represented by the relationships between concepts.

Figure 10 represents the structural metamodel of process activities. A process is an object composed by activities and an activity is an object with emphasis on operations (actions). An action is an elementary step of an activity and does not have externally visible structure. In the context of this paper, an action is not an object, but an operation of the activity object. Thus, a process represents a conceptual perspective and is composed by elementary steps that are activities. An activity represents an operational perspective and contains actions.

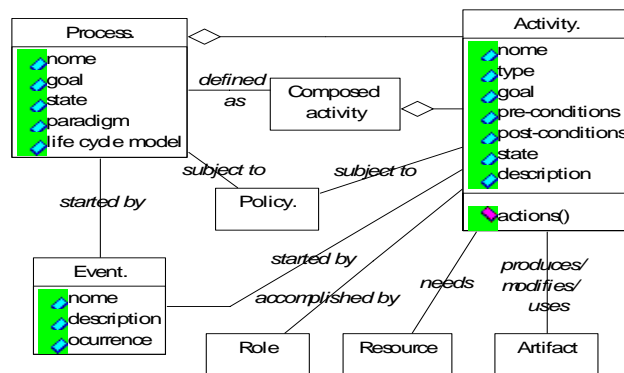


Figure 10. Activity structural metamodel

According to Figure 10, processes have a name that identifies them, a goal to guide the activity goal, a state, a paradigm that refers to the approach to develop a system, such as object orientation, and a life cycle model that refers to how the process phases are accomplished, such as iterative and incremental. A process is subject to policies and controls and is composed by interrelated activities.

A composed activity is defined as a process and is composed by a set of simple or composed activities. A composed activity can be, for example, a sub-process or a process phase.

A simple activity has name, a type to categorize the activity, goal, pre- and post-conditions, state and description. The name identifies the activity. The type allows activities to be categorized. Pre- and post-conditions indicate the conditions that must be evaluated before and after the activity is carried out. The description contains the information about the use of an activity, allowing the reuse of experiences and knowledge related to its accomplishment.

Events refer to the stimulus that processes and activities must respond to. Events can have periodic occurrence, asynchronous control or result from the verification of conditions. Name identifies events. Description can store the event restrictions. Occurrence contains the logical conditions that need to be fulfilled for the event to occur. Conditions can be related, for example, to time or to the existence of resources.

Figure 11 presents the software development process obtained from the metamodel of Figure 10. Figure 11 exemplifies the use of metamodels to define models, and activity attributes and operations. Comparing this figure with Figure 10, software development process is process; requirement, design, implementation and test phases are composed activities; and activity to elicit requirement is activity.

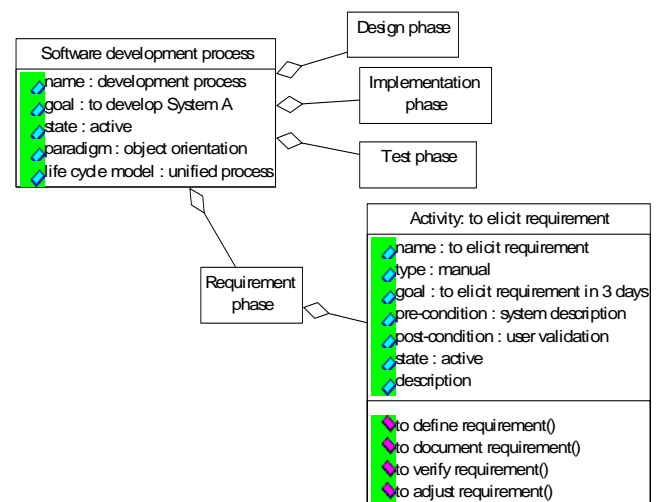


Figure 11. Example of activity structural model

Figure 12 presents the model states of the process. The states depicted in this figure are generic. Specific states depend on the process characteristics and its domain and context.

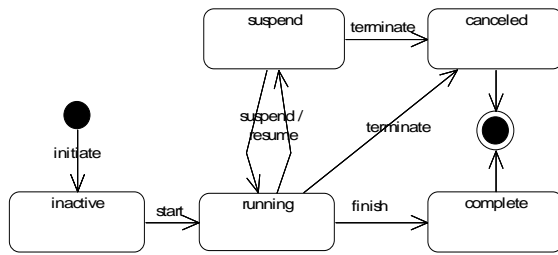


Figure 12. Model of process states

In inactive state the process instance was created and expects the occurrence of an event to start its execution. A process in the running state has one or more instances of activities and the process iterates through all active activities. In suspended state the process instance expects some condition to be canceled or to continue its execution. In canceled state the process instance is finished before all process activities are completed. In completed state the process execution was completed and the post-conditions are evaluated.

Figure 13 is an example of the control and data flow depicted in BPMN.

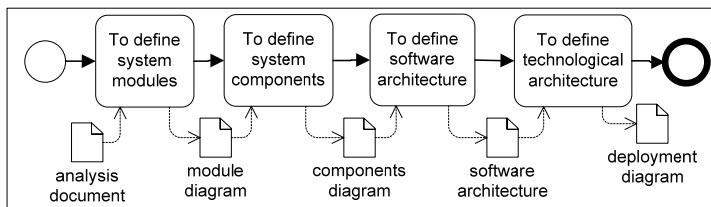


Figure 13. Model of control and data flow

The process depicted in Figure 13 contains the activities that compose the process used to define software architecture.

### Final considerations

The main contribution of this paper is to represent a process as an object. This can be used to define metamodels and models of process and their elements as objects.

The proposal in this paper makes it possible to deal with process modeling in the same way to analyze and design object oriented software systems.

The metamodels and models produced can be used to define tools to depict and to automate process enactment.

### References

- [1] Booch, G. (1991): Object oriented design with applications, Benjamin/Cummings.
- [2] Taylor, D.A. (2003): Engenharia de negócios com tecnologia de objetos, Axcel Books.
- [3] OMG - Object Management Group (2005): UML - Unified modeling language: superstructure, version 2.0.
- [4] Lazovik, A. Aiello M., and Papazoglou, M. (2004): Associating assertions with business processes and monitoring their execution. In II International Conference on Service Oriented Computing, pp. 94-104.

[5] Lindsay, A. Downs, D., and Lunn, K. (2003): Business processes - attempts to find a definition. Information and Software Technology, vol. 45, pp. 1015-1019.

[6] ABNT. Associação Brasileira de Normas Técnicas (1999): NBR ISO/IEC 12207:1998. Tecnologia de informação - Processos de ciclo de vida de software, Rio de Janeiro: ABNT.

[7] Wangenheim, C.G.V., et al. (2006): Experiences on establishing software processes in small companies, Information and Software Technology, vol. 48, n. 9, p. 890-900.

[8] Pressman, R.S. (2002): Engenharia de software, 5ª ed., Rio Janeiro: McGraw-Hill.

[9] Lepasaar, M., and Makinen, T. (2002): Integrating software process assessment models using a process meta model. In IEMC'02 - Engineering Management Conference, IEEE International, vol. 1, pp. 224-229.

[10] ISO/IEC - International Standardization Organization/International Engineering Consortium (1996): ISO/IEC 10746-3: open distributed processing - reference model - part 3: architecture.

[11] Bertollo, G., and Falbo, R.A. (2003): Apoio automatizado à definição de processos em níveis. In II Simpósio Brasileiro de Qualidade de Software, pp. 77-91.

[12] Shlaer, S., and Mellor, S.J. (1992): Object lifecycles. Modeling the world in status, Yourdon Press.

[13] Rumbaugh, J., et al. (1997): Modelagem e projeto baseado em objeto, Rio de Janeiro: Campus.

[14] Jacobson, I., et al. (1998): Object-oriented software engineering, Addison Wesley.

[15] Booch, G., Rumbaugh, J., and Jacobson, I. (1999): The unified modeling language user guide, Addison Wesley.

[16] Barnett, W.A., et al. (1994): An architecture for the virtual enterprise. In 1994 IEEE International Conference on Systems, Man, and Cybernetics, vol. 1, pp. 506-511.

[17] Holt, J. (2006): No views in bad news. Process modelling within a framework. In IEE Seminar on Process Modelling Using UML, pp. 27-46.

[18] Caetano, A., Silva, A.R. and Tribolet, J. (2005): Using roles and business objects to model and understand business processes. In 2005 ACM Symposium on Applied Computing, ACM Press, pp. 1308-1313.

[19] OMG - Object Management Group (2005): SPEM - Software process engineering metamodeling specification, version 1.1.

[20] Kammer, P.J. (2000): Supporting dynamic distributed work processes with a component and event based approach. In 22nd International Conference on Software Engineering, ACM Press, pp. 710-712.

[21] White, S.A. (2004): Business process management notation, version 1.0.