A Process Engineering Method based on a Process Domain Model and Patterns

Charlotte Hug, Agnès Front, Dominique Rieu

LIG, SIGMA Team – Grenoble University
220 rue de la Chimie - BP 53
38041 Grenoble Cedex 9 - FRANCE
{Charlotte.Hug, Agnes.Front, Dominique.Rieu}@imag.fr

Abstract. There are many different process meta-models that offer different viewpoints of a same process: activity oriented, product oriented, decision oriented, context oriented and strategy oriented. However, the complementarity between their concepts is not explicit and there is no consensus about the concepts themselves. This problem leads to inadequate process meta-models with organization needs, so the instantiated models do not correspond to the specific demands and constraints of the organizations or projects. However,, method engineers should be able to build process meta-models according to the specific organization needs. We propose a method to build unified, fitted and multi-viewpoints process meta-models. The process engineering method is based on a process domain model and on patterns.

Keywords: process meta-model, process model, information system engineering, method.

1 Introduction

There are many different process meta-models and each of them represents a different viewpoint of the same process [1], [2], [3]: activity oriented, product oriented, decision oriented, context oriented and strategy oriented, but there is no explicit mapping between these viewpoints. Thus, it is difficult for a method engineer to choose the best process meta-models, and, furthermore, to grasp their complementarity. Moreover, most of the process meta-models do not provide any extension mechanism and therefore are harder to adapt to the requirements.

To introduce the issue of this paper, let us present the situation of a method engineer who works in a small business dedicated to information system engineering. His objective is to build a model to represent the process of information system engineering. The developer team works using the method eXtreme Programming – XP [4] which is an activity oriented approach. The team would like to add a goal oriented approach to the XP method, the method engineer thus has to define a new process meta-model. The method engineer has to build a model representing:

– The goals and subgoals, with the KAOS formalism (requirement 1) [5],

– The phases (that are part of the XP lifecycle) broken into activities which are executed by actors, with the formalism of SPEM [6] use case diagram (requirement 2),

– The products created during activities, with the SPEM activity diagram (requirement 3).

These three requirements allow us to expose the issue this paper tries to solve. The method engineer needs a model that represents different process viewpoints. Different meta-models allow representing each of these viewpoints. There is a multitude of existing process meta-models, each of them representing a particular viewpoint of the process without explicit mapping between them. However, to create a unique process meta-model, there should be explicit correspondence between the meta-models viewpoints. Currently, these correspondences do not exist, therefore to represent different viewpoints, a method engineer needs to build as much process meta-models as viewpoints. Furthermore, the existing process meta-models do not propose extension mechanisms to be customized to meet the requirements of method engineers (except SPEM).

Method engineers need to build multi viewpoints process meta-models, that are unified in only one meta-model, and that can be customized to meet their requirements.

In this paper, we propose a method to help building unified, fitted and multi-viewpoints process meta-models. Our proposition consists of a process engineering method based on a process domain model and patterns, that allows building unified, fitted and multi-viewpoints process meta-models according to the organizations or method engineer's needs. These process meta-models can then be instantiated and executed as processes, in concordance with the project specificities.

The method is based on a process domain model presented in section 2. The first phase of the method consists of selecting the concepts from the process domain model that meet the requirements of the method engineer to build the process meta-model; this phase is presented in section 3. Then, method engineers should use design patterns or business patterns to refine the process meta-model. Section 4 presents the patterns and section 5 explains the refinement phase. Section 6 presents the instantiation of the process meta-model. Finally, section 7 presents the related works and section 8 concludes this paper.

2 Process domain model

The proposed process domain model contains the main concepts of existing process meta-models. It is a high-level domain model, which does not include secondary concepts (for example state product) of the main concepts (for example product). The secondary concepts will be integrated further in the method in order to simplify and lighten the construction of the process meta-model. The domain model is composed of two different abstraction levels: the intentional abstraction level, which represents the goals, the objectives of a process and the operational abstraction level, which represents the actions to concretize these objectives. The domain model

comprises different viewpoints or modelling axis of a process. A viewpoint is a process perspective; it is not necessarily associated to a particular actor or role as other viewpoint definitions [7], [8].Let us briefly describe the different concepts of the process domain model presented in Fig. 1.

Operational and intentional levels are represented as stereotypes. We use different kinds of graphical links to distinguish the different associations between concepts. A classic association represents an association between two concepts in the same abstraction level. For example, Work Unit and Role are both at the operational level; a classic association links them. The dashed lines with an arrow represent the concretization of one concept of the intentional level into another concept of the operational level. For example, a Work Unit concretizes a Strategy.

The concept Work Unit represents something that is done during the process. A Work Unit has conditions, creates (out), uses (in) or modifies (in/out) Work Products, and raises new Issues. This concept comes from activity oriented process meta-models such as SPEM [6], Open Process Framework [10], OOSPICE [11] and SMSDM [12], which concentrate on the activities performed in producing a product and their ordering [1].

A Work Product is something produced or used, during the process, that can be a deliverable (the software for example). The Work Product concept proceeds from product oriented process meta-models, as the State Transition which is a ViewPoint template presented in [8], the Statecharts meta-model [13], the Entity meta-model [14], and the Statemachine meta-model [9]. Product oriented process models couple the product state to the activities that generate this state [1].

A Role does something during the process. A Role carries out a Work Unit, is responsible for a Work Product and can select alternatives to issues. This concept comes from activity oriented process meta-models.

Issues are problems rising during the execution of a process. When an Issue appears, some alternatives respond to it. An argument can cite work product(s) to object or support an alternative, and then to contribute to the advance of a Work Unit. Issue, Alternative and Argument concepts come from decision oriented process meta-models such as CAD° (Conversation among Agents on Decisions over Objects) of the DAIDA project [15], inspired from Potts and Brun [16], and IBIS [17]. Decision oriented process models present the successive transformations of a product or elicitations due to decisions [1].

A Context is composed of a Situation and an Intention of an actor at a given moment of a project. The Intention is a goal, an objective that the application engineer has in mind at a given point of time [18]. The Situation represents the part of the product undergoing the process [19], a Condition concretizes a situation as well as a Work Product concretizes an Intention. The European project NATURE introduced the notion of context, which defined a meta-model of the same name [20] The context oriented process models consider the situation and the intention of an actor (analyst, method engineer…) at a given moment of the project [1].

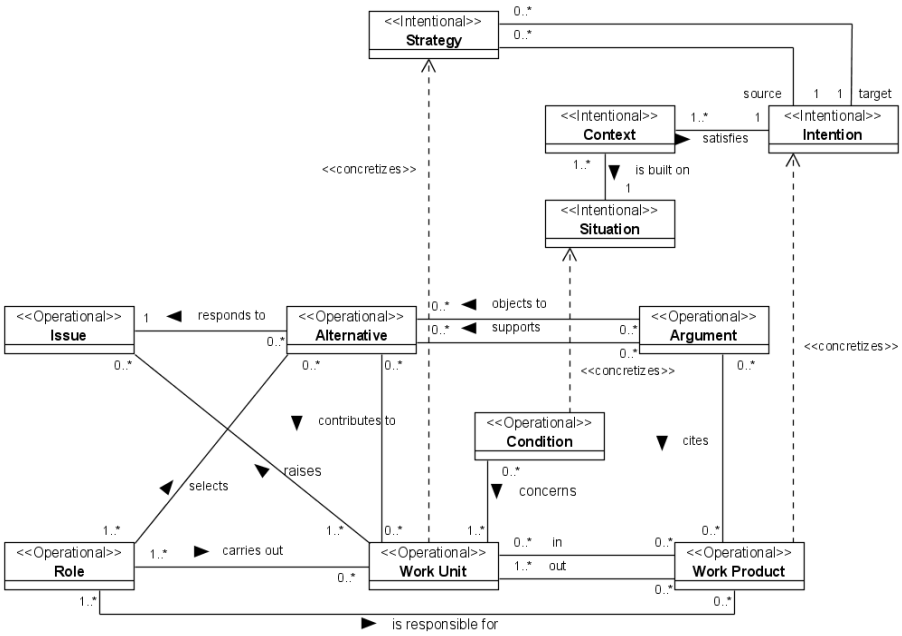

Fig. 1. The process domain model.

At last, a Strategy is an approach, a manner to achieve an Intention. It allows joining a source intention to a target intention. Work Unit concretizes Strategy. The strategy concept comes from the strategy oriented process models that allow representing multi-approach processes and plan different possible ways to elaborate the product basing on intention and strategy notions [18]. As far as we know, MAP [18] is the only strategy oriented process meta-model to date. This meta-model allows representing intentions that can be achieved according to different strategies.

This concise process domain model includes the core concepts of existing process meta-model and method engineering experts will be lead up to enrich it.

Different abstraction levels and viewpoints compose the model: **Table 1** sums up for each concept, which are its viewpoint and its abstraction level. Activity, product and decision viewpoints use the concepts of Role, WorkProduct and Work Unit. Strategy and Context viewpoints use the Intention concept. For example, to represent an activity viewpoint, which is at the operational abstraction level, we would need the concepts role, work product, work unit and condition.

Table 1. Abstraction levels, viewpoints and concepts.

Abstraction level	Viewpoint	Concept	
Intentional	Strategy	Strategy	Intention
	Context	Context	
		Situation	
Operational	Decision	Issue	Role, Work Product, Work unit
		Alternative	
		Argument	
	Product		
	Activity	Condition	

Some concepts of the process domain model cannot be separated from other concepts. Their existence depends on other concepts existence. **Table 2** presents the depender concepts and their dependee concepts. For example, an alternative cannot exist without an issue, but an issue can exist without an alternative. Some concepts compulsorily depend on more than one concept: a context cannot exist without a situation and an intention. Other concepts depend on at least one concept; for example, a role can depend on a work unit, an alternative or a work product. The cardinalities in the process domain model partly represent these constraints.

Table 2. The depender concepts and their dependee concepts.

Depender	Dependee
Strategy	{Source Intention $\wedge$ Target Intention}
Context	{Situation $\wedge$ Intention}
Argument	Alternative
Alternative	Issue
Condition	Work Unit
Role	{Alternative $\vee$ Work Unit $\vee$ Work Product}

The next section presents the selection phase that is based on the process domain model.

3 Selection phase

During the selection phase, the method engineer chooses the needed concepts from the process domain model to produce a process meta-model. This process meta-model contains the main concepts and will be refined in the next phase of the method. The method guides the method engineer choosing the concepts thanks to a questionnaire. Table 3 presents an extract of this questionnaire. A concept of the process model domain corresponds to each answer about the Information System

Engineering (ISE) process. In order to help the method engineer answering the questions, he can visualize synonyms, alias or examples of each concept.

Table 3. Extract of the questionnaire.

Question	Synonyms, also known as, examples	Concept
Do you need to represent actions that are executed during the ISE process?	Activity, phase, task, work definition	Work unit
Do you need to represent something that is produced, used or modified during the ISE process?	Product, document, model, software, program	Work product
Do you need to represent someone that carries out an action during the ISE process?	Actor, developer, analyst, system	Role
Do you need to represent goals or objectives of the ISE process?	Objective, goal, subgoal	Intention

When choosing a concept, the dependee concepts and the associations between them are "imported" in the process meta-model. The dependee strategy partially ensures the integrity of the process meta-model each time a new concept is added from the process domain model. Nevertheless, our objective is not to check the consistency and the integrity of the process meta-model. This task is not in the scope of our research.

Let us go back to the method engineer's problem. Thanks to the questionnaire, he can choose the concepts that meet his requirements. For example, to meet the requirement 1, he will choose the concept of intention. The concepts work unit, role meet the requirement 2, and the concept of work product met the requirement 3. Some of the imported associations are useful: the associations "in" and "out" allow representing work products that are used or produced during a work unit. The association "carries out" permits representing that a role carries out a work unit. The method engineer then obtains the process meta-model presented in Fig. 2.

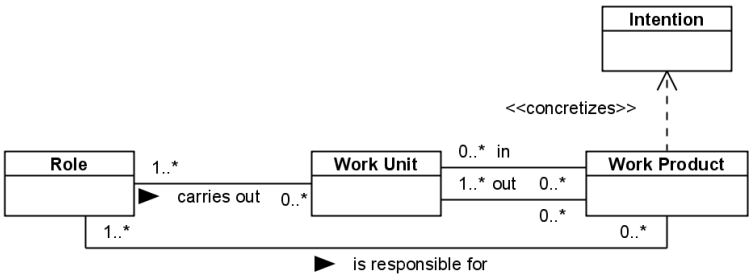

Fig. 2. The process meta-model.

However the requirements are partially met: the process meta-model does not allow representing subgoals (requirement 1) and composition of work units (requirement 2). The association "is responsible for" which was imported automatically does not correspond to any requirement. The process meta-model obtained is thus incomplete, the next phase will help the method engineer to refine it, using design and business patterns.

4 Design and business patterns

The refinement phase consists of refining the process meta-model using meta-modelling techniques formalized in design patterns and business patterns. Design patterns describe a frequently occurring problem in a context and a general repeatable solution that resolves it. Method engineers can reuse design patterns to enrich the process meta-model. Many design patterns already exist, but they still have to be adapted for process meta-modelling. In [3] we present the "Concept-Concept Category" pattern, a pattern for process meta-modelling. This design pattern allows the categorization of concepts with specific properties and sharing common properties at two modelling levels (see Fig.3). For example, the pattern allows defining work units such as "write user stories" and work unit categories such as "activity". Each work unit belongs to a work unit category. Moreover, some properties of the process meta-model can be instantiated at the model level to express common characteristics of projects, and other properties can be instantiated at the process level to express specificities of a particular project. For example, the method engineer needs to define properties for the activity "Write user stories" of XP method in general as the name of the activity and if it is optional or not, and properties of the execution of this activity in a particular project as the length or the starting and ending dates of the activity.

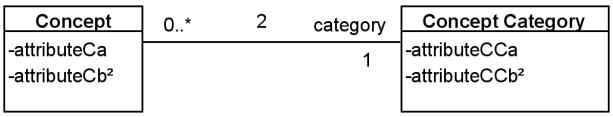

Fig.3. The concept-Concept Category pattern.

Business patterns represent process meta-model fragments. Process meta-model fragments are part of existing process meta-models that method engineers can reuse to detail one or more concepts of the process meta-model developing secondary concepts. For example, Fig. 4 represents a fragment of the product oriented process meta-model State-Transition [8]. We can use this fragment when we need to represent the different state of a work product.

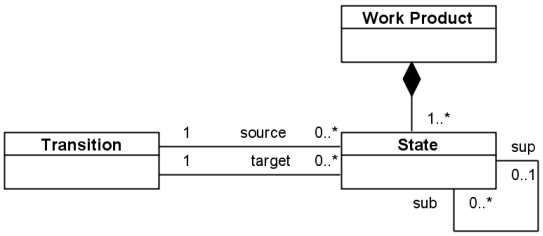

Fig. 4. Fragment of the State-Transition process meta-model.

A pattern system composed of process patterns (the process of the method) and product patterns (design and business patterns), allows representing our process engineering method in order to standardize their representation. We do not present the pattern system in this paper because of lack of space. All the patterns are represented in the formalism P-SIGMA [21], a common formalism for patterns representation that allows the clarification of the patterns selection interface and facilitates the organization of pattern systems. We also dispose of a tool, AGAP [21], a development environment for defining and using patterns. This tool is used as a repository for the design patterns, the business patterns (fragments) and the process patterns.

5 Refinement phase

During the refinement phase, the method engineer selects from the process meta-model the concepts he wants to enrich. A list of appropriate patterns is proposed for each concept. This list is constituted from reports of experts and measures that will be described in a forthcoming paper. According to different strategies, the method engineer chooses to reuse a particular pattern by the problem it resolves, by its frequency of use and/or by its adequacy with the chosen concept. If no existing pattern satisfies him, he can complete the list of appropriate patterns adding a new one. This new pattern has to be validated later by experts, so that other engineers could reuse it. The method engineer can add or delete associations, aggregations or compositions between concepts. The method engineer can choose to continue the improvement of the process meta-model as long as he needs to.

Let us come back to our example. In section 3, the method engineer has created an incomplete process meta-model. He can now complete it carrying out these actions (see Fig. 5):

– The method engineer applies the design pattern “Concept-Concept Category” [3] to the concept work unit because he needs to distinguish a phase from an activity, and he needs to define properties about an activity in general and properties of the execution of an activity in a particular project.

– He applies a reflexive composition to intention, work unit and work unit category, to meet completely the requirements 1 and 2.

– He creates the association “precedes-follows” for work unit and work unit category to allow sequences of work unit and work unit categories.

– He creates the association “parallel” for work unit and work unit category to allow the execution in parallel of work units and work unit categories.

He deletes the association “is responsible for” which is useless in his case.

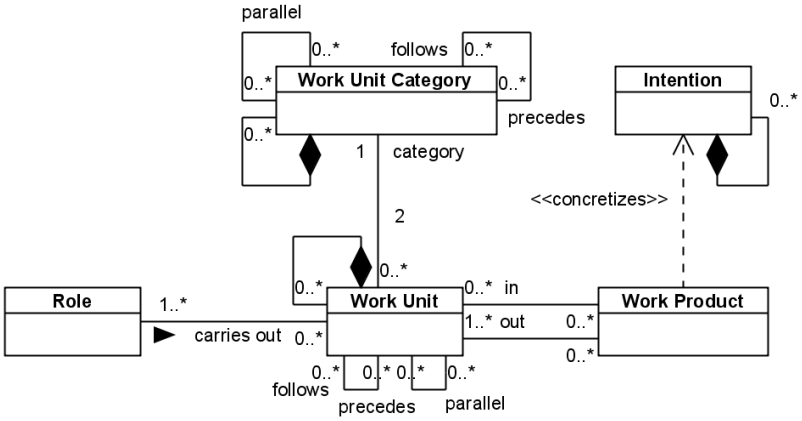

Fig. 5. The final process meta-model.

The method engineer can now instantiate this process meta-model.

6 Instantiation

In this section, we present an extract of the process model instantiated by the method engineer from the process meta-model.

The method engineer wants to define intentions and sub-intentions of the information system engineering process as “determine the requirements” decomposed into “A brief description of what the customer wants” and “A specification of what the customer wants”. He instantiates these intentions and sub-intentions from the concept Intention, of the process meta-model. The two sub-intentions are concretized in the XP method respectively by the work products “User stories “and “Requirements”. These work products are created by the activities “Write user stories” and “Define requirements” that are carried out by the developer and the customer. The work product “User stories” is used by the activity “Define requirements”. These activities are executed in sequence and they are part of the exploration phase. The exploration phase is part of the XP lifecycle.

Because of lack of space, we do not present the attributes in the figure.

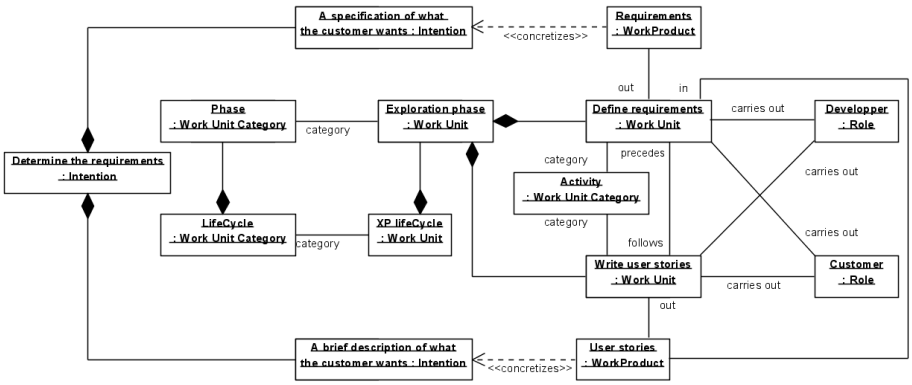

Fig. 6. Extract of the process model of the XP method.

The extract of the process model in Fig. 6 is:
– Multi viewpoints: it contains goal oriented and activity oriented viewpoints,
– Unified: the viewpoints are represented in the same model,
– Fitted to the method engineer requirements.

The model in Fig. 6 is represented as an object diagram. The objects can be represented in the desired formalisms. The intentions and sub-intentions can be represented with the KAOS formalism [5]. The phases, the activities, the roles and the links between them can be represented as a use case diagram. At last, the activities, the work products and the links between them can be represented as an activity diagram.

7 Related works

In this section, we discuss some works related to information system engineering process. [22] present a process reuse architecture. This architecture allows storing, classifying and retrieving process frameworks, process patterns or usual processes. Our solution is different because we provide a method to build process meta-models, whereas the process reuse architecture allows building process models.

[23] provide a meta-model to define process patterns to build and improve process models. This solution focuses on process models while our solution focuses on process meta-models. However, some mechanisms could be adapted to process meta-modelling (pattern searching, selecting, etc.).

[24] presents an ontology for information system development (ISD). This ontology aims to help understanding ISD, analyzing and comparing ISD artefacts and supporting the creation of new ISD artefacts. It is a low-level ontology and no method is provided to help building information systems using the ontology. This ontology comprises different domains: action (activity oriented), actor, object

(product oriented) and purpose (decision and goal oriented). It does not include intentional level as strategy and context.

At last, [25] present a process meta-model for software development methodologies and their enactment. This process meta-model comprises producers, work products, work units and stages. There are no decision, strategy and intention viewpoints.

8 Conclusion

This paper presents a process domain model, which main concepts come from different types of existing process meta-models. This model is the base of a process engineering method that helps building unified, fitted and multi-viewpoints process meta-models for information system engineering. The method is composed of two phases: the first phase, selection, allows the method engineer choosing the different needed concepts from the domain model to build a first version of the process meta-model. It is refined during the second phase, refinement, mainly using patterns. Then, the method engineer can instantiate the process meta-model, according to the needs of the organization, the project or the method (XP, RUP…).

In the future, the process domain model is lead up to be enriched and the method has to be validated, rebuilding existing process meta-models and models using case studies. Once the method is validated, we will then have to implement a tool that will easily guide the users in the construction of their process meta-models: the two phases, selection and refinement, could be assisted by a workflow.

References

1. Rolland, C.: A comprehensive view of process engineering. In Pernici, B., Thanos, C. (Eds.) CAiSE'98, LNCS, vol. 1413, pp. 1--24. Springer, Heidelberg (1998)
2. Rolland C.:L'ingénierie des méthodes : une visite guidée. e-TI - la revue électronique des technologies d'information. Premier Numéro, 25 octobre (2005) http://www.revue-eti.netdocument.php?id=726
3. Hug, C., Front, A., Rieu, D.: Ingénierie des processus : une approche à base de patrons. INFORSID, 471--486 (2007)
4. Wells D., 17th February (2006), http://www.extremeprogramming.org
5. Cediti SA: A KAOS tutorial. 5th September (2003)
6. OMG: Software Process Engineering Metamodel Specification. Version 1.1 (2005)
7. Sommerville, I., Kotonya, G., Viller, S., Sawyer, P.: Process Viewpoints. In Schäfer, W., (Ed.) EWSPT 1995, LNCS, vol. 913, pp 2--8. Springer, London (1995)
8. Finkelstein, A., Kramer, J., Goedicke, M.: ViewPoint oriented software development. In third International. Workshop on Software Engineering and Its Applications, pp. 374--384 (1990)
9. OMG: Unified Modeling Language: Superstructure. Version 2.1.1 (2007)
10. Open Process Framework, http://www.opfro.org, 16th December (2005)
11. OOSPICE, Software Process Improvement and Capability Determination for Object-Oriented/ Component-Based Software Development, www.oospice.com, 7 October (2002)

12. Australian Standard: Standard Metamodel for Software Development Methodologies, AS 4651-- 2004 (2004)
13. Harel, D.: Statecharts: A Visual Formulation for Complex Systems, *Science of Computer Programming*, vol. 8, n° 3, 231--274 (1987)
14. Humphrey, W.S., Kellner, M. I.: Software Process Modeling: Principles of Entity Process Models. In *ICSE* 1989, *IEEE Computer Society / ACM Press*, pp. 331--342 (1989)
15. Jarke, M., Mylopoulos, J., Schmidt, J.W., Vassiliou, Y.: DAIDA: An Environment for Evolving Information Systems. *ACM Transactions on Information System*s, vol. 10, n° 1, 1--50 (1992)
16. Potts, C., Bruns, G.: Recording the Reasons for Design Decisions. In ICSE 1988, IEEE Computer Society Press, pp. 418--427 (1988)
17. Kunz, W., and Rittel, H.W.J.: Issues as elements of information systems. Working Paper n° 131, Heidelberg-Berkeley (1970)
18. Rolland, C., Prakash, N., Benjamen, A.: A Multi-Model View of Process Modelling. Requirements Engineering, vol. 4, n°4, 169--187 (1999)
19. Plihon, V., Rolland, C.: Modelling Ways-of-Working. In Iivari, J., Lyytinen, K., Rossi, M. (Eds) *CAiSE'95*, *LNCS*, vol. 932, pp 126--139. Springer-Verlag, London(1995)
20. Rolland, C., Souveyet, C., Moreno, M.: An Approach for defining ways-of-working. *Information System Journal*, vol. 20, n° 4, 337--359 (1995)
21. Conte, A., Fredj, M., Hassine, I., Giraudin, J.-P., Rieu, D.: A Tool and a Formalism to Design and Apply Patterns. In Bellahsene, Z., Patel, D., Rolland, C. (Eds.) OOIS, LNCS, vol. 2425, pp. 135--146. Springer (2002)
22. Fiorini S.T., Do Prado Leite J.C.S., De Lucena C.J.P.: Process Reuse Architecture. CAiSE'01, vol. 2068, pp. 284--298, Springer (2001)
23. Tran H. N., Coulette B., Dong B. T.: Modeling Process Patterns and Their Application. *ICSEA'07*, *IEEE Computer Society*, pp. 15--20 (2007)
24. Leppanen M.: Towards an Ontology for Information Systems Development. In EMMSAD'06, pp. 36--374. University Press, Namur, Belgium (2006)
25. Henderson-Sellers B., Gonzalez-Perez C.: A comparison of four process metamodels and the creation of a new generic standard. Information & Software Technology. 47, 49--65 (2005)