A method to build information systems engineering process metamodels

Charlotte Hug, Agnès Front, Dominique Rieu, Brian Henderson-Sellers

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

A Method to build Information Systems Engineering Process Metamodels

Charlotte Hug*, Agnès Front*, Dominique Rieu*, Brian Henderson-Sellers**

*LIG – SIGMA, Grenoble University, 220 rue de la Chimie, 38400 Saint Martin d'Hères, France
Charlotte.Hug@imag.fr
Tel: (33) 4 76 63 55 79
Fax: (33) 4 76 63 55 50

**Department of Software Engineering, Faculty of Information Technology, University of Technology, Sydney, P.O. Box 123, Broadway NSW 2007, Australia

Abstract: Several process metamodels exist. Each of them presents a different viewpoint of the same information systems engineering process. However, there are no existing correspondences between them. We propose a method to build unified, fitted and multi-viewpoint process metamodels for information systems engineering. Our method is based on a process domain metamodel that contains the main concepts of information systems engineering process field. This process domain metamodel helps selecting the needed metamodel concepts for a particular situational context. Our method is also based on patterns to refine the process metamodel. The process metamodel can then be instantiated according to the organisation's needs. The resulting method is represented as a pattern system.

Keywords: information systems engineering, method engineering, process metamodel, method, pattern.

1. Introduction

Many information systems engineering method definitions exist. According to Harmsen (Harmsen, 1997) "an Information Systems Engineering Method is an integrated collection of procedures, techniques, product descriptions, and tools, for effective, efficient, and consistent support of the IS engineering process" where, according to Booch (Booch, 1991), a method is "a rigorous process to generate a set of models describing various aspects of software being built using some well-defined notation". Integration and rigour can best be appropriated by the use of a metamodel that defines the modelling language (to describe the work products) together with a metamodel describing process and producer elements and integrating with the modelling language (e.g. ISO/IEC, 2007). The former metamodel is used to ensure the correctness of the work product models and diagrams whilst the latter metamodel is used to underpin the process model (a.k.a. methodology). Process models offer approaches for developing information systems in a well-disciplined manner. Here, we will consider process modelling in the context of information systems engineering.
An information systems engineering process is a complex endeavour. (Cauvet, 2006) presents the various arguments for the modelling of an engineering process. It is argued that the process must be modelled in order to depict its various stages, as well as to guide the different actors involved throughout the life cycle. The modelling of a process may also allow the subsequent monitoring of its progress in real time, that is to say its implementation, often called its enactment. Finally, the model allows the recording of what has been done, for reusing the most generic parts or for monitoring and improving the process. Organisations wishing to model their process — as accurately as possible and at all stages of the engineering life cycle — are faced with various problems. For example, the organisationally-agreed process model or methodology should be a good fit to the organisational maturity, skills level etc. (Henderson-Sellers and Nguyen, 2004). Creating such a methodology is the focus of situational method engineering (SME) (Brinkkemper, 1996). The definition of a process model as might be created using SME should be headed by concepts, rules and relationships; a process metamodel is therefore necessary for instantiating process models. Indeed, the need for industry adoption of metamodels for solving business problems, perhaps using a model-driven engineering (MDE) approach has been recently urged by Forrest (Forrest, 2008) based on his first-hand industry involvement.
Many different process metamodels exist ((Harel, 1987), (OMG, 2005), (OPF, 2005), (Potts and Bruns, 1988), (Rolland et al., 1995), and (Rolland et al., 1999) among others); while there are others, such as (AS, 2004), (OOSPICE, 2002) that include process issues as part of an all-embracing software development methodology. Each of them defines different concepts:
- (OMG, 2005), (OPF, 2005) deal with the concepts of work unit, work definition and roles,
- (Harel, 1987) comprises the concepts of product, state and transition,

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

- (Potts and Bruns, 1988) includes the concepts of issue, alternative and argument,
- (Rolland et al., 1995) deals with the concepts of situation and intention,
- (Rolland et al., 1999) include the concepts of intention and strategy.

Each of these different process metamodels represents a different viewpoint of the information systems engineering process. A viewpoint is a process perspective; it is not necessarily associated with a particular actor or role. (Dowson, 1987), (Mi and Scacchi, 1996) and (Rolland, 1998) define the different process metamodel categories, which we consider here as viewpoints. Based on this categorization, we can say that, overall, (OMG, 2005) and (OPF, 2005) are activity-oriented process metamodels, (Harel, 1987) is a product-oriented process metamodel, (Potts and Bruns, 1988) is decision-oriented, (Rolland et al., 1995) is context-oriented and (Rolland et al., 1999) is strategy-oriented. There is obviously a link between these different process metamodels since they represent the same conceptualization but from a different viewpoint. However, these links have never been explicitly defined – a focus of our paper. The correspondences between the different viewpoints should be explicit in order to provide a complete vision of the information systems engineering process. Nowadays, to define a multi-viewpoint information systems engineering process, different process metamodels of the needed viewpoints have to be used, effectively independently, and instantiated since no formalized links exist to show the correspondences between the viewpoints.

Furthermore, some of the existing process metamodels are so large that they can be difficult to understand. Large metamodels thus tend to be only partially used by most process/method engineers. Hence, existing CAME tools based on big metamodels may waste resources.

For very many individual projects, one of these predefined metamodels will be an appropriate fit to the organizational requirements. Indeed, this is the reason that standards, such as those of ISO, exist. However, in many other instances, method engineers need to combine concepts from two or more metamodels i.e. they need:

- Multi-viewpoints and unified process metamodels,
- Metamodels fitted to the organisation or project specificities.

(Karagiannis and Kühn, 2002) advocate the introduction of flexibility into metamodels. Consequently, in this paper, we propose a method that supports the building of fitted, multi-viewpoint and unified process metamodels in order to support the real-life situations when a predetermined, standardized metamodel is inadequate. To parallel SME but at a higher abstraction level, we name this approach as Situational MetaModel Engineering (SMME). This method is based on a process domain metamodel that contains the main concepts of the different existing process metamodels such as work unit, role, strategy etc. The first phase of the method consists of selecting the needed concepts from the process domain metamodel. A draft process metamodel containing the main concepts of the required process metamodel is produced. The draft process metamodel is then refined and extended in the second phase of the method, mainly using patterns. The method is represented thanks to a pattern system.

The paper is organized as follow. Section 2 presents the different existing process metamodels, a brief synthesis and the main problems encountered. Section 3 describes the phases of the proposed method, selection and refinement, and the resources used: the process domain metamodel and the patterns. Section 4 presents the pattern formalism and the pattern system and Section 5 describes an example. Section 6 discusses the related works and Section 7 concludes this paper.

2. Existing process metamodels

In this section, we present the different existing process metamodels classified according to the different process viewpoints: activity, product, decision, context and strategy.

2.1. Activity-oriented process metamodels

Activity-oriented process metamodels allow building models concentrating on the activities and tasks performed in producing a product together with their ordering (Rolland, 1998). They typically comprise the concepts of Work Unit or Work Definition that have Products as inputs and outputs and are performed by a Role.

SPEM (OMG, 2005), The Open Process Framework (OPF, 2005), OOSPICE (OOSPICE, 2002), SMSDM (AS, 2004) and ISO 24744 (ISO, 2007) are process metamodels, mainly activity-oriented. Some of them, such as SPEM and ISO 24744, include other viewpoints such as product varying degrees of detail.

The process models of methods such as RUP (Kruchten, 2000), XP (Beck, 1999) and SCRUM (Schwaber and Beedle, 2001) are instances of activity-oriented process metamodels and thus are activity-oriented process models.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

Figure 1 presents an extract of SPEM metamodel (OMG, 2005). A WorkDefinition can be an Activity, a Phase, an Iteration or a Lifecycle (the last two not shown). WorkDefinition describes the work performed in the Process. A ProcessRole defines responsibilities over specific WorkProducts, and defines the roles that perform and assist in specific activities. ProcessRole is the subclass of ProcessPerformer (not shown in Figure 1). A ProcessPerformer defines a performer for a set of WorkDefinitions in a process (OMG, 2005). The ActivityParameter meta-class allows the specification of the inputs and outputs (WorkProduct) of a WorkDefinition.

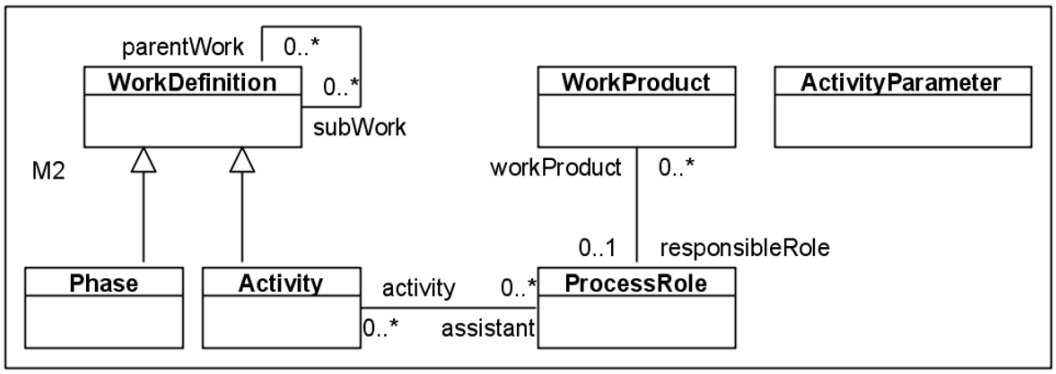


Figure 1. The activity-oriented process metamodel SPEM.

2.2. Product-oriented process metamodels

Product-oriented process metamodels permit the instantiation of models that couple the product state to the activities that generate this state (Rolland, 1998). They present the concepts of Product that have different States and transitions defined between the states. The product state represents the situation of a product at a precise moment of the process, the transitions being defined between these states to represent the order in which the states can change. The transitions are relations between a source state and a target state; they are triggered by an event.
The metamodel for Statecharts (Harel, 1987), State Machines (OMG, 2007), as well as the metamodel of the Entity Process Model (EPM) (Humphrey and Kellner, 1989) and the State-Transition template (Finkelstein et al., 1990), are examples of product-oriented process metamodels that use the above concepts.
Figure 2 shows the simplified product-oriented process metamodel of the State-Transition approach (Finkelstein et al., 1990) as an example of this class of process metamodels. One or more States composed a Product, while Transitions are defined between the States.

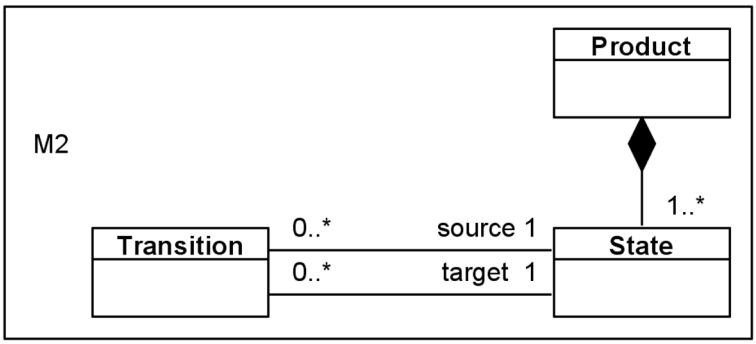


Figure 2. Simplified State-Transition product-oriented process metamodel.

2.3. Decision-oriented process metamodels

Decision-oriented process metamodels allow building models that present the successive transformations of a product due to decisions (Rolland, 1998). They count the concepts of Issues that need answers defined as Alternatives. Alternatives can be supported or objected by Arguments. An issue is a problem met during the information systems process engineering. Alternatives are different options to sole the issue. The alternatives are based on arguments.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

IBIS (Kunz and Rittel, 1970) first introduced the notion of decision-oriented process. It was then improved by Potts and Bruns (Potts and Bruns, 1988), Potts (Potts, 1989) and in the DAIDA project (Jarke et al., 1992). Figure 3 presents the decision-oriented process metamodel defined in (Potts, 1989). Steps during information systems engineering can raise Issues. Alternatives respond to Issues and are supported or objected by Arguments, citing Artefacts. Alternatives can contribute to Steps. An Issue can review an Artefact that can be modified by a Step.

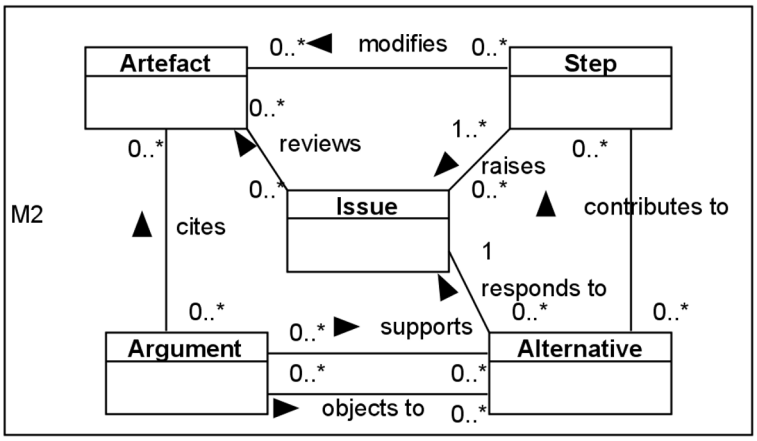


Figure 3. Potts decision-oriented process metamodel.

2.4. Context-oriented process metamodels

Context-oriented process metamodels allow building models representing the situation and the intention of an actor at a given moment of the project (Rolland, 1998). The couple of Situation plus Intention forms a Context. The key concepts of this kind of metamodel are the Context that is composed of a Situation and an Intention. The situation is a part of a product under design that is the object of a decision. The intention represents the objective, i.e. the goal that an actor wants to achieve according to the situation (Plihon, 1996).
The notion of context was first defined in (Grosz and Rolland, 1990) and finalized within the context-oriented process metamodel NATURE (Rolland et al., 1995), in a European project of the same name. (Rolland et al., 2000) adapted the NATURE metamodel to Enterprise Knowledge Management. The concept of intention had replaced the concept of decision, which has also been done in the NATURE metamodel. It is also worth noting that Situation and Intention are the main concepts of Situational Method Engineering, as defined in (Brinkkemper, 1996), as SME consist of building methods “on the fly” according to the method defined at a precise moment (Situation) and what method engineers need to add in the method (Intention).
Figure 4 shows an extract of the NATURE context-oriented process metamodel (Rolland et al., 1995). A Context is composed of a Situation (in respect to a product) and an Intention (an objective in respect to the product). There are three types of Context. The Plan based Context is composed of ordered Contexts. The Choice-based Context corresponds to a Situation that requires the exploration of Alternative contexts that can be based on arguments. Executive-based Contexts implement the Intention into an Action that transforms a part of a product.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

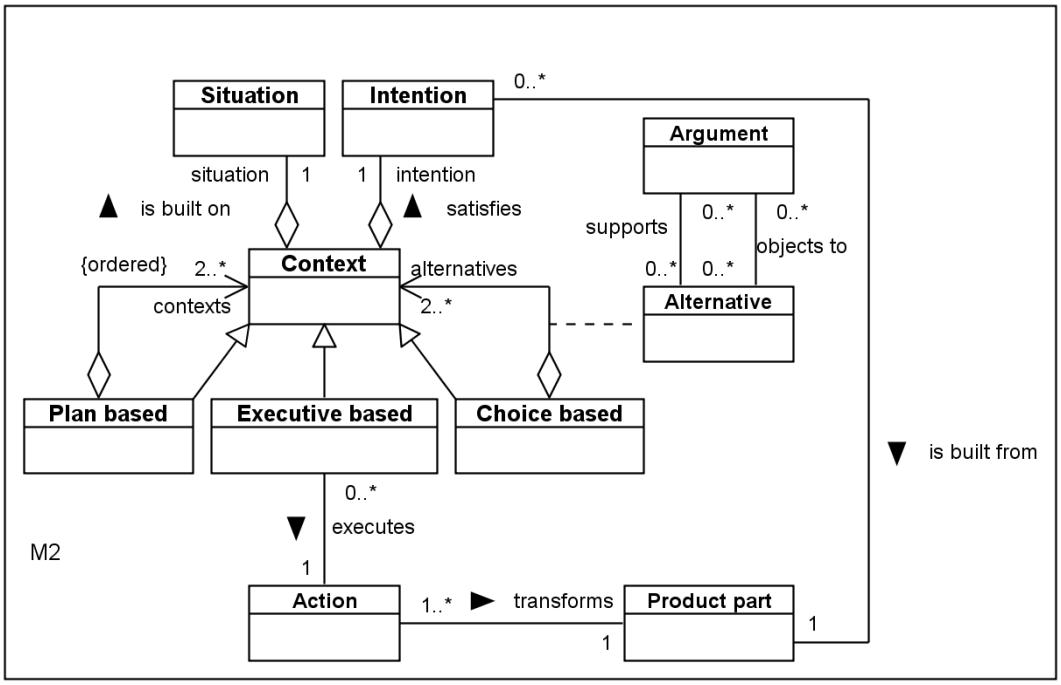


Figure 4. Extract of the NATURE context-oriented process metamodel.

2.5. Strategy-oriented process metamodels

Strategy-oriented process metamodels allow building models representing multi-approach processes and plan different possible ways to elaborate the product based on the notion of intention and strategy (Rolland et al., 1999). Strategy and Intention are the main concepts of this kind of metamodels. An intention is a goal, an objective to achieve. A strategy is a manner to achieve the intention (Zoukar, 2005). [As far as we know, MAP (Rolland et al., 1999) is the only strategy-oriented process metamodel published to date, although a goal-focussed SME approach for process model construction is described in (Gonzalez-Perez et al., 2008) and the work product pool approach of (Gonzalez-Perez and Henderson-Sellers, 2007) is also loosely related in that context.]
The process model of the requirement engineering method “CREW-l’Écritoire” (Rolland et al., 1999) has been formalised using MAP. MAP has also been used to represent an engineering method for matching ERP functionalities and organisational requirements (Zoukar and Salinesi, 2004).
Figure 5 presents the simplified strategy-oriented process metamodel for the MAP approach. A Section is composed of a Strategy, a Source Intention and a Target Intention. A MAP is composed of one or more Sections. It always has a Start Intention and a Stop Intention.

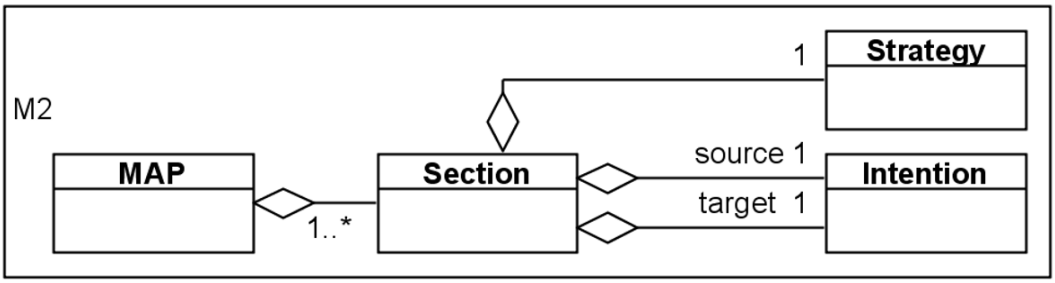


Figure 5. Extract of the strategy-oriented process metamodel MAP.

2.6. Synthesis

Table 1 synthesizes the concepts of the most representative process metamodels. We do not present all the concepts of every process metamodel but only the more significant. [In (Henderson-Sellers and Gonzalez-Perez, 2005), a similar but more limited comparison was done between the activity-oriented process metamodels of the

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

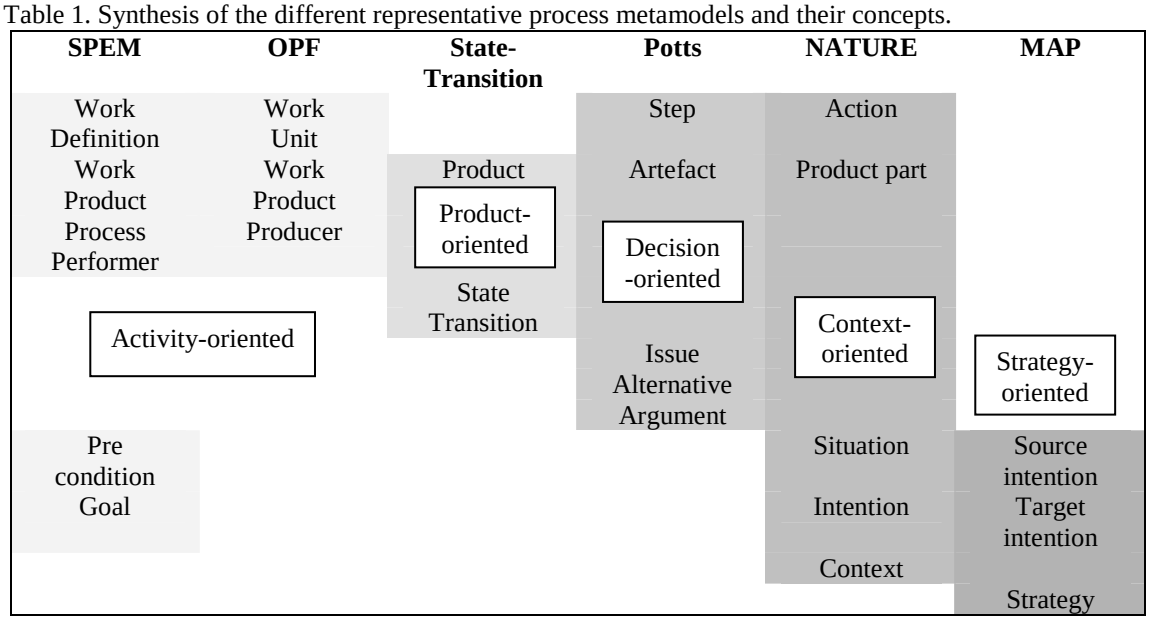
SMSDM, SPEM, OOSPICE, OPF and LiveNet. Table 1 includes two activity oriented process metamodels, SPEM and OPF, but also product oriented process metamodels, decision, context and strategy oriented process metamodels.] It can be seen that some concepts are common to various process metamodels, for example WorkDefinition, but these are frequently not at the same granularity level, e.g. WorkDefinition and WorkUnit are very general terms whereas Step and Action are very concrete. Almost all the process metamodels deal with the concept of Product, but only the State-Transition metamodel (and all the product-oriented process metamodels) focuses on States. In contrast, the concepts of Issue, Alternative and Argument are only defined in decision-oriented process metamodels.

Some concepts, such as Precondition and Goal, seem very similar to Situation and Intention or Source Intention and Target Intention as they all correspond to the state of the process before or after an action, respectively; although they would appear to be located at different levels of abstraction. Pre-condition and Goal (a.k.a. Post-condition) are concrete, while Situation and Intention are more abstract.

The concept of Context is only present in the context-oriented process metamodel NATURE and the concept of Strategy only exists in the strategy-oriented process metamodel.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

Table 1. Synthesis of the different representative process metamodels and their concepts.



2.7. Problems

The first problem met is the mere existence of existing process metamodels, whatever the viewpoint. How can a method engineer, easily and quickly, choose the process metamodel corresponding to the organisation's needs? There is no existing framework to guide the method engineer through the selection of the most appropriate process metamodel. There are only frameworks that guide them in the selection of concepts from the framework process metamodel (OPF, 2005) for the OPF, Eclipse Process Framework (EPF, 2006) for SPEM and Xome (Gonzalez-Perez, 2005) for the SMSDM and a subset of OOSPICE. Therefore, there is a separate framework for each one process metamodel.

Moreover, there is only a partial consensus on the vocabulary used in the different process metamodels. (Henderson-Sellers and Gonzalez-Perez, 2005) have presented a synthesis of the different concepts used in activity-oriented process metamodels and their correspondence. This comparison shows that the same word can be used to represent different concepts from one metamodel to another. For example, an "Activity" in SPEM does not correspond to an "Activity": in OPF. These differences can easily lead to difficulty in understanding the different process metamodels for method engineers, who need operational and comprehensible process metamodels quickly.

Furthermore, the vocabulary and concepts used are different from one organisation to another, because of the enterprise culture, knowledge and practices collected over the years. However, existing process metamodels are not adaptable to the organisation's vocabulary. When using a process metamodel, why should the organisation not use its own vocabulary and concepts instead of being forced to use the existing process metamodel terms?

The existing process metamodels only propose one viewpoint and there are no correspondences between the concepts of process metamodels of different viewpoints. If the method engineer needs to represent many viewpoints, he/she can only use each metamodel by itself, and does not have the correspondences between their concepts.

Finally, the majority of the process metamodels do not propose extension or adaptation mechanism except SPEM (OMG, 2005) and ISO 24744 (Henderson-Sellers and Gonzalez-Perez, 2006b). The method engineers cannot adapt the process metamodels needed to the organisational requirements.

The objective of this paper is to allow method engineers to create:

- Unified,
- Multi-viewpoint,
- Fitted process metamodels.

The term "unified process metamodel" means that all the concepts that the method engineers need to specify are grouped in only one metamodel, whatever the viewpoints they specify. If the process metamodels are unified, they are multi-viewpoint. All the viewpoints could be represented in the same process metamodel. Finally, the

process metamodels will be fitted to the organisational requirements. The method engineers will specify the vocabulary they want to use, the concepts and the associations between these concepts.
To do this, we propose that method engineers build themselves the process metamodel(s) that the organisation needs, following a two-phase method as described in the next section.

3. The method

In this section, we present the two phases of our method (Hug et al., 2008) and the resources made available to method engineers. The first phase, selection, consists of choosing the main required concepts from the process domain metamodel provided. The second phase, refinement, allows method engineers to refine, complete and extend the draft process metamodel obtained in the first phase, by applying metamodelling patterns and business patterns.

3.1. The process domain metamodel

The process domain metamodel, visualized in Figure 6, comprises the main concepts from the existing process metamodels of different viewpoints.
The concept WorkUnit is the main concept of an activity-oriented process metamodel. A WorkUnit can own Conditions and can be carried out by a Role. WorkProduct comes from a product-oriented process metamodel; however, we do not propose the concepts of State and Transition in the process domain metamodel because they are too detailed at this stage: these concepts are additional detail to the basic concept of WorkProduct. The concepts of Issue, Alternative and Argument come from the decision-oriented process metamodels. Based on the comparisons made in Table 1, we have easily linked these concepts to WorkUnit and WorkProduct. Context, Intention and Situation come from context-oriented process metamodels. The concept of Intention is linked to the Strategy concept, which comes from strategy-oriented process metamodels.

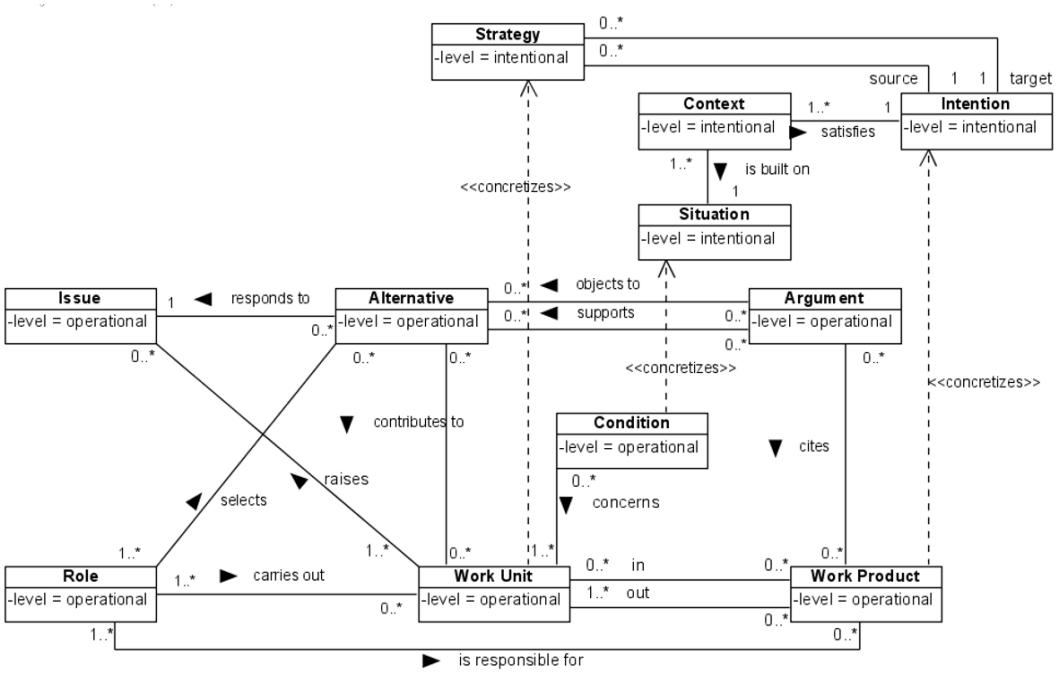


Figure 6. The process domain metamodel.

Furthermore, we introduce two levels of abstraction. These levels of abstraction are not linked to the abstraction levels of the OMG but correspond to a more or less concrete way by which to view the process. The intentional abstraction level represents the goals and objectives of an ISE process, while the operational abstraction level represents the actions needed to concretize these objectives. These levels are useful if ISE processes are defined in two steps: in the first step, the main objectives are defined, and in the second step, the objectives are detailed and described with operational terms. The levels can also be useful if two different kinds of actors participate in the ISE process since any stakeholders of the business field tend to think in terms of goals of the organisation

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

whilst information systems providers think in terms of actions. The abstraction levels are represented in the process domain metamodel as valued attributes that can be either intentional or operational. Figure 7 represents the concepts of the process domain metamodel placed according to their viewpoint and abstraction level.

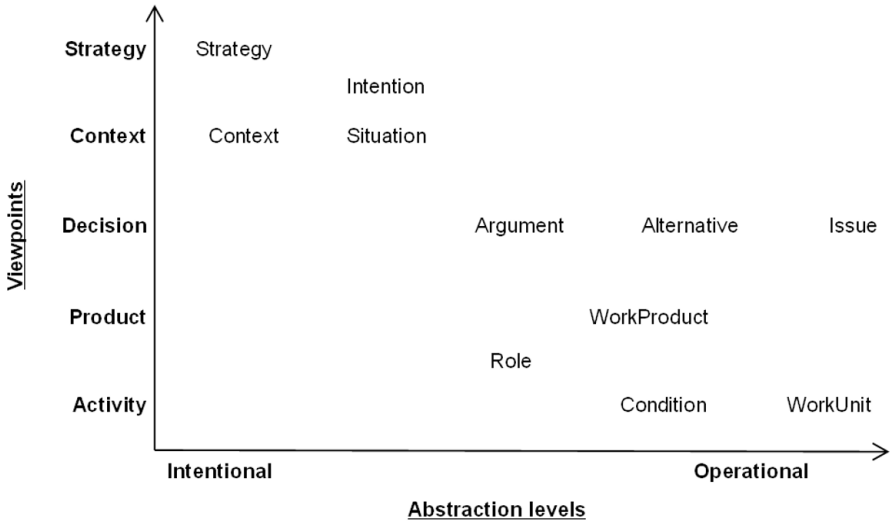


Figure 7. Concepts, viewpoints and abstractions levels.

3.2. The selection phase

During the selection phase, the method engineer completes a questionnaire (Table 2) – similarly to the approach of (Henderson-Sellers and Nguyen, 2004) in selecting method fragments to build the process model. The answers to the questionnaire will help to determine which concepts are included in the draft process metamodel and which excluded. For example, if the method engineer answers "yes" to the first question, the draft process metamodel will include the Intention concept. Each question of the questionnaire corresponds to a concept of the process domain metamodel.

Table 2. The questionnaire

Question	Synonyms, also known as, examples	Concept
Do you need to represent goals or objectives of the ISE[1] process?	Objective, goal, subgoal	Intention
Do you need to represent how an intention is achieved?	Tactics, approach, manner	Strategy
Do you need to represent the situation of an actor at a given moment of the ISE process?	Circumstance	Situation
Do you need to represent both intention and situation?		Context
Do you need to describe problems encountered during the ISE process?	Problem, toughness, question, difficulty	Issue
Do you need to represent answers to an issue?	Answer, choice, possibility, contingency, option, dilemma	Alternative
Do you need to represent an argument, a proof to object or support an alternative?	Proof, reason	Argument
Do you need to represent something that is produced, used or modified	Product, document, model, software, program	Work product

[1] Information Systems Engineering

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

during the ISE process?		
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 actions that are executed during the ISE process?	Activity, phase, task, work definition	Work unit
Do you need to represent condition on the action?	Pre-condition, post-condition, constraint	Condition

When the method engineers have answered all the questions, they obtain a draft process metamodel. Each time a concept is integrated in the draft process metamodel, the existing associations of the process domain metamodel are imported. If the method engineer chooses the concept of WorkUnit and then of WorkProduct; the associations "in" and "out" would be automatically imported into the draft metamodel.
Moreover, some concepts of the process domain metamodel cannot be separated from other concepts i.e. there is an existence dependency between two elements in the process metamodel. Table 3 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.
When the method engineer chooses a dependee concept, the depender concept is automatically added into the draft process metamodel, in order to ensure that this constraint is met.

Table 3. 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}

In order to facilitate the selection of concepts, the method proposes associated concepts to the method engineer when selecting one concept. For example, when the method engineer chooses the WorkUnit concept, the method will propose him/her the concepts of WorkProduct, Role, Condition, Issue, Alternative and Strategy. This allows the selection phase to be less irksome, notably answering all the questions of the questionnaire.
Figure 8 represents the described above process followed during the selection phase. It is represented as a MAP, using the MAP formalism of (Rolland et al., 1999). Nodes represent Intentions and edges represent Strategies. A MAP always begins with the "Start" Intention and Stops with the "stop" Intention. Each intention of the MAP corresponds to a step of the selection phase. The first strategy ("Yes answer") is the only way to integrate a concept in the draft process metamodel. The "Dependee strategy" has to be selected if there is a dependee concept; if not the "Associated concept strategy" or "Association strategy" can be selected. The "Associated concept strategy" permits integrating the concepts that are linked with the previously selected concepts while the "Association strategy" allows integrating the associations between the concepts in the draft process metamodel. The "Improvement strategy" will allow method engineers to improve their draft process metamodels. The "Completion strategy" is used when the draft process metamodel is complete. The strategies between the intentions show that not every step is necessary, for example, "Integrate dependee concepts" and "Integrate associated concepts".

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

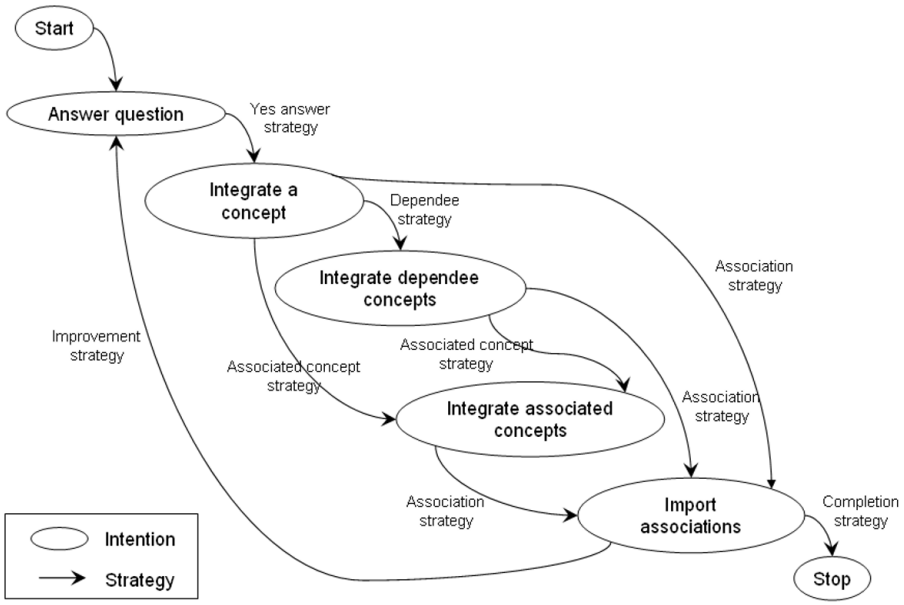


Figure 8. The selection phase.

3.3. The patterns

In this subsection, we present the resources used in the second phase of the method. These resources are patterns of two types: metamodelling patterns and business patterns.

3.3.1. Metamodelling patterns

A metamodelling pattern has the same purpose as a design pattern, in the sense that a design pattern describes a frequently occurring problem in a context and a general repeatable solution to resolve it (Alexander, 1979). We seek the same characteristics, but as a contribution to metamodelling rather than design (Karagiannis and Kühn, 2002).
Many design patterns already exist, such as the Object-Oriented Patterns of Coad (Coad, 1992), the GoF Design Patterns (Gamma et al., 1995) etc. These patterns are intended to be reused at the modelling level. Here, we seek analogous patterns that can be used at the metamodelling level. We begin by studying the published design patterns to determine if they can be reused in some way at the metamodelling level. Then, they will need to be adapted to metamodelling.
Some metamodelling patterns have already been defined. The MP workshop of the French-speaking conference IDM aims to define a metamodelling patterns catalogue (MP, 2007) although it should be noted that these patterns do not concern process metamodelling.
The Powertype (Odell, 1994) can be used as a metamodelling pattern. This pattern allows the instantiation of metaclasses into objects that also inherit properties of another metaclass. (Gonzalez-Perez and Henderson-Sellers, 2006b)'s approach is based on the Powertype pattern to build a process meta-model for software development methodologies. The Powertype is therefore fitted for metamodelling.
(Hug et al., 2007) present a metamodelling pattern, the purpose of which is similar to the Powertype pattern. The "Concept-Concept Category" pattern allows the categorization of specific concepts and the instantiation of properties at two modelling levels. This pattern is based on the "Item-Description" pattern (Coad, 1992) and on the Deep Instantiation idea of (Atkinson and Kühne, 2001). It is thus an adaptation of the "Item-Description" pattern to metamodelling. Figure 9 presents the "Concept-Concept Category" metamodelling pattern. The superscript "2" on the attribute name represents the potency of the Deep Instantiation to indicate that the attribute or association will be instantiated at the second modelling level.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

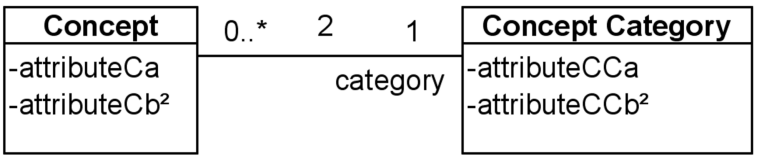


Figure 9. The “Concept-Concept Category” metamodelling pattern.

Figure 10 shows the reuse of the “Concept-Concept Category” pattern for the Work Unit concept. The Work Unit and Work Unit Category concepts are part of the meta-model. General attributes such as the name or the description of Work Unit and Work Unit Category are instantiated in the process models because they are valid for all the process models defined within the organisation. Specific attributes as the date of beginning or the manager of the phase are instantiated at the enactment of the process.

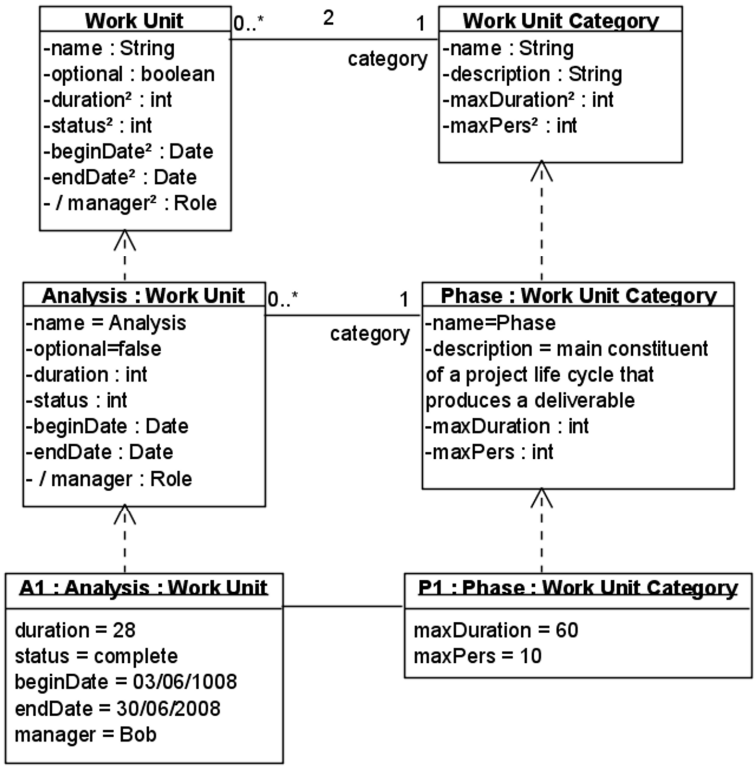


Figure 10. Reuse of the "Concept-Concept Category" pattern for the Work Unit concept.

3.3.2. Business patterns

Business patterns represent process metamodel fragments. Process metamodel fragments are part of existing process metamodels that method engineers can reuse in order to detail one or more concepts of the process metamodel developing secondary concepts. They are generic specifications to create solutions suitable for a given application problem in the field, here information systems engineering processes metamodelling. For example, we may consider Figure 2 as a process metamodel fragment called “State-transition” business pattern. The major concept of this fragment is Product (equivalent to WorkProduct) of which the process metamodel domain is comprised. If the method engineer wants to develop the product-oriented viewpoint of their process metamodel, they can reuse the “State-transition” business pattern. The State and Transition concepts will be integrated in their draft process metamodel, as well as the associations between the concepts Product, State and Transition.
Any process metamodels presented in Section 2 can be regarded as a business pattern.

3.4. The refinement phase

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

During the refinement phase, the method engineer selects the concepts they need to enrich in order to meet the organisational requirements. For each concept, they can choose to reuse an existing metamodelling pattern or a business pattern. They can choose a pattern based on the problem it resolves, to its adequacy or the frequency of its use:
- The problem the pattern resolves is one of the items describing it. The paper describes later the pattern formalism used.
- The adequacy corresponds to a subjective measure that the method engineers give for reusing a pattern for a particular concept. This measure is given when the process metamodels have been regularly used.
- The frequency of use is a statistical measure that calculates the ratio of the number of times a pattern P_i has been used for a concept C_j divided by the number of times any pattern has been used for the concept C_j. The measure is represented in Equation 1.

$$\frac{R(P_i, C_j)}{R(\sum P_i, C_j)}$$

Equation 1. The frequency of use measure.

The process followed by the method engineer in the refinement phase is presented in Figure 11 as a MAP. The method engineers select the concept they want to enrich; they can reuse a pattern or create a new pattern that will be validated later by experts so that it could be use by other method engineers. The method engineers restart the process as many times as necessary.

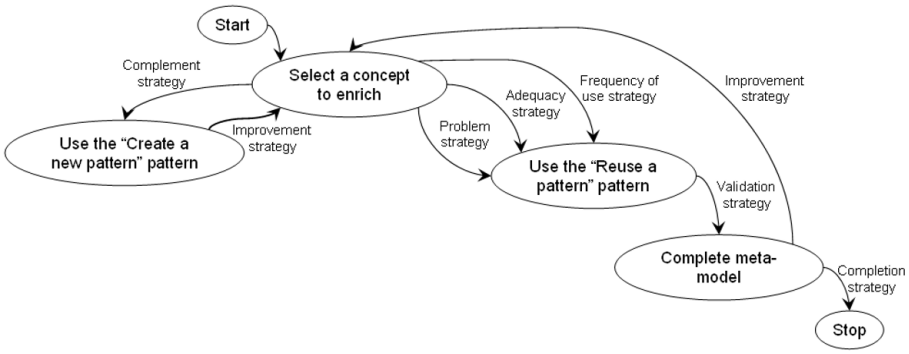


Figure 11. The refinement phase.

4. The method representation

In order to represent metamodelling patterns and business patterns homogeneously, we also choose to represent the method phases as process patterns. We use the P-Sigma formalism (Conte et al., 2002) to represent all the patterns. These patterns form a pattern system.

4.1. The P-Sigma formalism

P-Sigma is a formalism that allows the standardization of product and process representation. This formalism was introduced because of the increasing number of patterns libraries offering product or process patterns whose range and coverage are diversified (analysis, design or implementation patterns, general, domain or enterprise patterns). Such libraries, although complementary, are difficult to combine during application development. This is mainly because no common formalism for patterns representation exists. P-Sigma thus offers a way of expressing a semantics that is common to the majority of formalisms proposed in the literature, in order to make a uniform expression of product patterns and process patterns, to make explicit the pattern selection interface and to allow a better organization of patterns libraries (Conte et al., 2002).
In other words, the representation of the metamodelling patterns, business patterns and process patterns of the method is homogenous, which will ease their comprehension and selection. The P-Sigma formalism is composed of three parts that comprise many items (cf. Table 4).

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

Table 4. The P-Sigma formalism.

Pattern interface	
Identification	Defines the couple (problem, solution) that references the pattern.
Classification	Defines the pattern function through a collection of domain keywords (domain terms).
Context	Describes the pre-condition of pattern application.
Problem	Defines the problem solved by the pattern.
Force	Defines the pattern contributions through a collection of quality criteria associated to a technology.
Pattern realization	
Process Solution	Indicates the problem solution in terms of a process to follow. An activity diagram allows representing the process.
Model Solution	Describes the solution in terms of expected products (a class diagram and optionally a set of sequence diagram).
Application Case	Describes application examples of the Model Solution. This item is optional, but recommended in order to facilitate the understanding of the pattern solution.
Consequence	Gives the consequence induced by the pattern application.
Patterns Relations	
Uses, Refines, Requires, Alternative	

The pattern interface part comprises the items that help the selection of patterns. The pattern realization part gives the model solution or the process solution. The relation part describes the relationship with other patterns. Not all the items can be filled for every pattern, particularly process solution and model solution.

4.2. The pattern system

The pattern system comprises:
- Process patterns, to represent the method phases,
- Metamodelling patterns, to represent design patterns for metamodelling,
- And business patterns, to represent process metamodel fragments.

All these patterns are represented in the P-Sigma formalism.
We also use a tool, AGAP (Conte et al., 2002), a development environment that allows building and managing pattern formalisms and pattern systems. All the patterns of our method are stored in the AGAP pattern system. Table 5 presents the process pattern corresponding to the selection phase of the method, using the P-Sigma formalism.

Table 5. The Selection process pattern.

Identification	Selection
Classification	{process ^ phase ^ metamodel}
Context	This pattern does not need any pattern to be reused.
Problem	Helps choosing the needed concepts of the process domain metamodel, importing the dependee and associated concepts in order to obtain a draft process metamodel
Force	Guides the process through different strategies in order to obtain a draft process metamodel.
Process solution	Insert Figure8.tif near here 1. The first step consists of answering a question of the questionnaire. 2. If the answer is “yes”, the corresponding question concept is integrated in the draft process metamodel. 3. If the concept has a dependee concept, it is also integrated into the draft metamodel. 4. If there is any associated concept, the method engineers can choose whether or not to integrate them at that moment. 5. The associations between the concepts of the draft metamodel

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

are integrated from the process domain metamodel.
6. If the draft metamodel needs to be improved, the process restarts.
7. When the draft process metamodel comprises all the main concepts required, the process terminates.

5. Example

Let us present a situation met by a method engineer in a small business dedicated to information systems engineering. The development team is used to working with the eXtreme Programming method (Beck, 1999), which is an activity-oriented approach. The team would like to add a strategy-oriented approach; consequently, the method engineer has to build a new process metamodel representing:
- The intentions of the project and the strategies to achieve them.
- A life cycle composed of phases, themselves composed of activities. Activities are carried out by one or more roles and produce and/or use one product. The phases and the activities are linked with guards if necessary.
- The different product categories, such as text, diagram etc.
As the method engineer already knows, more or less, what is needed, the selection phase is easily accomplished. The questionnaire (Table 2) guides him/her through the selection of the concepts that correspond to the organisation's requirements.
The following concepts are imported from the process domain metamodel into the draft process metamodel: Strategy and Intention, WorkUnit, Role and WorkProduct. The associations and the link of concretization are automatically imported into the draft process metamodel, see Figure 12.

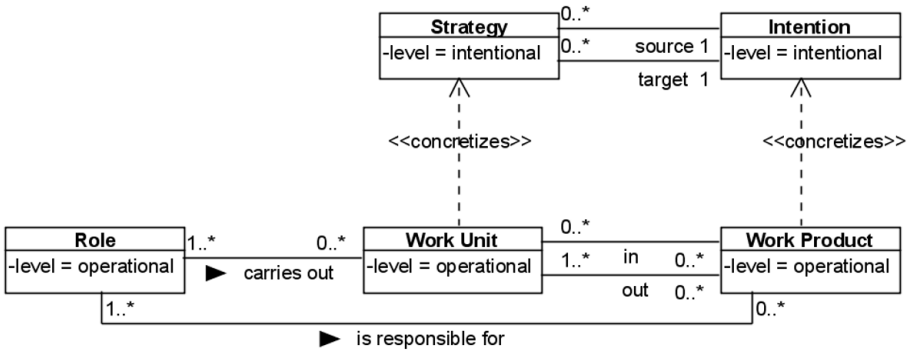


Figure 12. The draft process metamodel obtained in the selection phase.

As the process domain metamodel only contains the core concepts of information systems engineering process, it is easy and fast to understand it. Using the questionnaire, the method engineers do not manipulate directly the process domain metamodel: that facilitates the selection.
However, the draft process metamodel does not yet meet all the requirements since the following requirements cannot be represented:
- The sequence of strategies and intentions
- Distinguishing a life cycle from a phase etc.,
- The composition of the phase and activities,
- The sequence or work units with a guard,
- Distinguishing a document from a program.
The refinement phase allows the method engineer to complete the process metamodel to meet the missing requirements.
Firstly, to represent a sequence of strategies and intentions, the method engineer can reuse the MAP business pattern. This pattern is a fragment of the MAP process metamodel comprising the concepts of Strategy, Intention, Section and MAP, which allow the representation of a comprehensive sequence of strategies and intentions.
Then, to represent the composition and the sequence of work units, the method engineer can reuse the metamodelling patterns "Add a composition" and "Add a reflexive association" on the concept WorkUnit. The method engineer can add the guard to the reflexive association using the metamodelling pattern "Add an association class".

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

To allow distinguishing, for example, a work product that is a document from a work product that is a program, the method engineer needs to reuse the pattern "Concept-Concept Category" on the WorkProduct concept. In the same way, to distinguish a work unit that is a phase from a work unit that is an activity, the pattern "Concept-Concept Category" has to be reused on the WorkUnit concept.
The composition of work unit categories must also be modelled, to specify that activities compose a phase.
To distinguish a role that carries out an activity from a role that only assists this activity, the method engineer adds the association "assists" between the concepts of Role and WorkUnit.
As all the associations will be instantiated at the process execution level, the method engineer needs to specify it in the process meta-model. By applying deep instantiation (Atkinson and Kühne, 2001) on every association, the method engineer states the association will be instantiated into links at the process execution level and not at the process model level[2]. The superscripts "2" represent the deep instantiation.
The final process metamodel is obtained at the end of the refinement phase (see Figure 13).

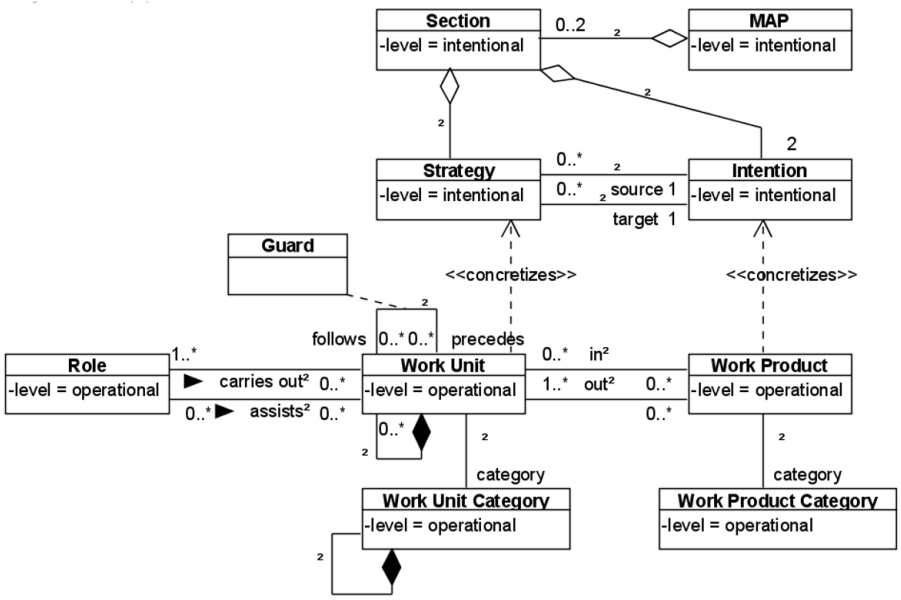


Figure 13. The final process metamodel obtained in the refinement phase.

The use of metamodelling patterns and business patterns give the method engineers wide possibilities of extension, refinement and adaptation. At the beginning of the refinement phase, method engineers manipulate a small process metamodel only containing the selected core concepts. It is easier to understand a reduced process metamodel and to refine it than understanding a big metamodel and removing the concepts that are not needed. The larger the metamodels are, the longer it is likely to take to understand all their concepts. Once the process metamodel has been completed, the method engineer can instantiate it to model the needed process model: XP and its strategy oriented viewpoint.
Figure 14 represents an extract of the process model instantiated from the process metamodel for the planning phase of XP.
The intention of the developers is to plan the releases; they can explore potential solutions if the estimates are uncertain. The intentions "Plan the releases" and "Explore potential solution" and the strategy "Uncertainty strategy" form a section; the sections together comprise the map.

The strategies "Uncertainty strategy" and "Certainty strategy" are respectively concretized by the work units "Create a spike solution" and "Create a release plan". These work units belong to the work unit category "Activity". These activities compose the phase "Planning phase", which is part of the XP lifecycle. The activities are performed by a developer, and assisted by a customer and produce a release plan (which concretizes the intention "Plan the releases") of schedule work product category, and a spike solution (which concretizes the intention "Explore potential solution") of the code work product category.

[2] A similar result is achievable using powertypes. Since elements in the model domain are clabjects, they can both represent associations in the model and also links in the endeavour domain where the process enactment takes place.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

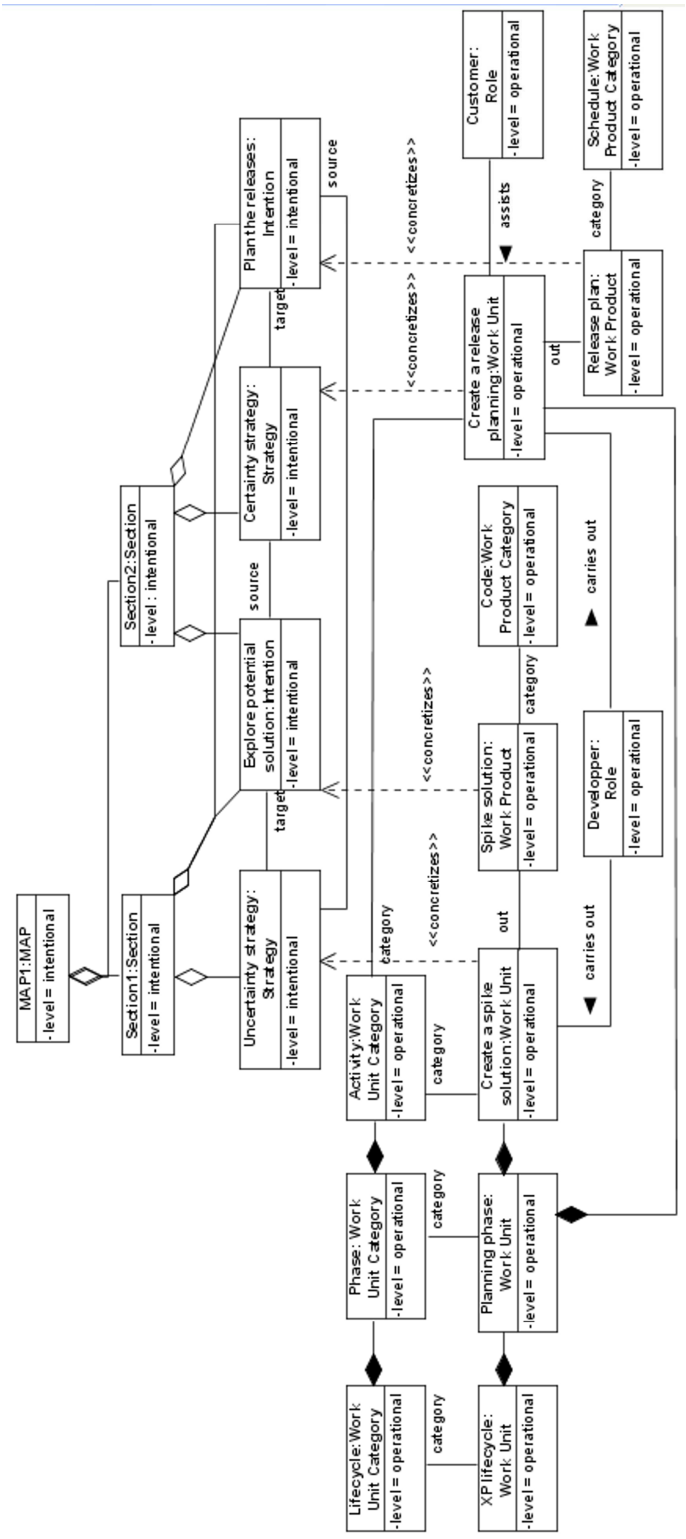


Figure 14. Extract of the process model.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

The process model clearly shows the correspondence between the strategy-oriented and the activity-oriented viewpoints. The method engineer could build the process model according to the organisation's needs, thanks to an adequate process metamodel.
The process model can also be represented in the desired formalism, for a better understanding, instead of being represented as a class diagram. For example, Figure 15 represents the strategy-oriented viewpoint of the process model using the MAP formalism while Figure 16 represents a part of the operational part of the process model using the use case formalism of SPEM and Figure 17 represents the extract of the operational part of the process model using the activity diagram formalism of SPEM. ISO/IEC 24744 (ISO, 2007) recommends several diagram types. For example, the phase and task information shown in Figure 16 would be depicted on a lifecycle diagram (Figure 18), while a high-level overview of how producers, work units and work products are inter-related can be shown on the new dependency diagram.

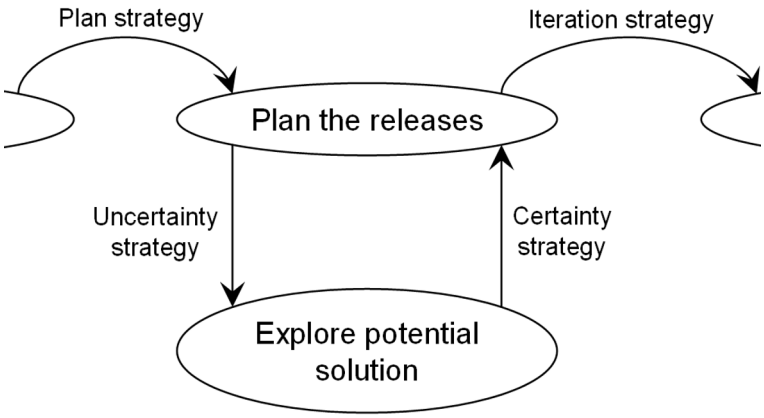


Figure 15. Extract of the intentional part of the process model.

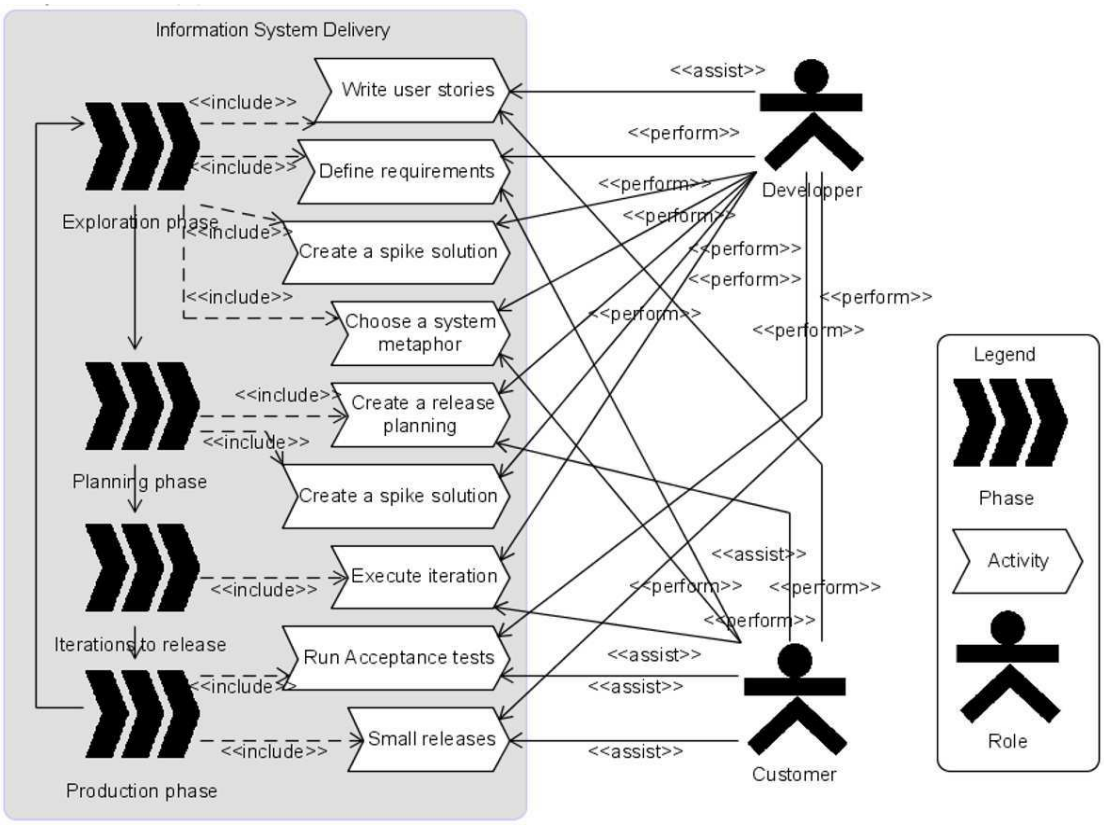


Figure 16. Operational part of the process model.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

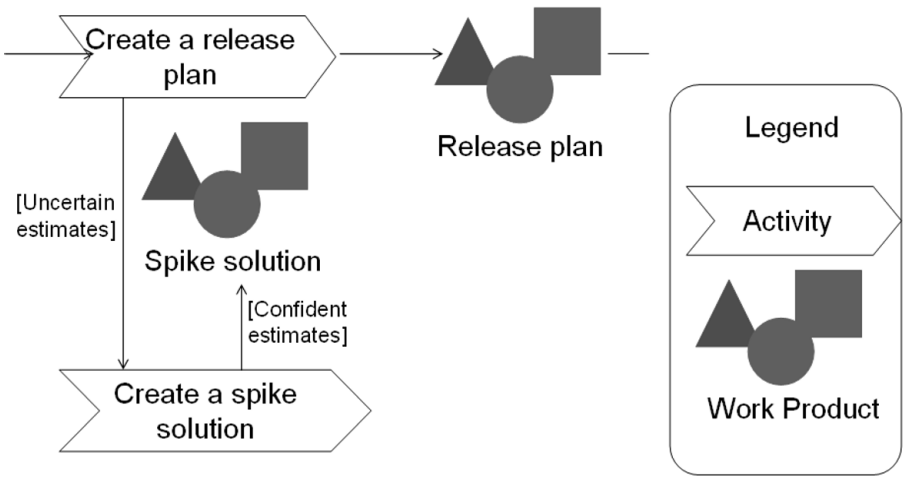


Figure 17. Extract of the operational part of the process model.

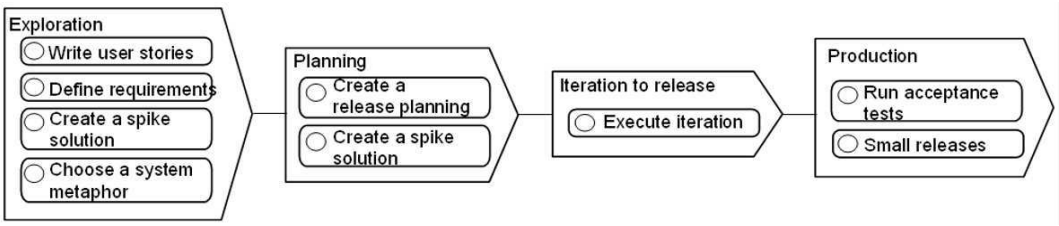


Figure 18. Phases and activities from Figure 16 expressed using ISO/IEC 24744 notation.

Our method presents a number of advantages:
- The cost of maintaining the process domain metamodel is minimal. An expert in process engineering, information systems engineering can maintain it thanks to technology watch in these domains. The maintenance of the process domain metamodel is easy, as it only contains a few concepts. Then, the impact of adding a new concept in the process domain metamodel will only consists of adding a few associations between the new concept and the older one.
- The cost of maintaining the pattern system is minimal. The users (method engineers) will add their own patterns that experts will validate and the users will naturally enrich the catalogue of patterns. Any users can use the validated patterns of the catalogue.
- The cost of education for the method users (method engineers) is also minimal. The process domain metamodel is small and its concepts are ordinary. The use of metamodelling patterns and business patterns might not be easy for beginners but pattern modelling is a common practice and it has often been shown that their use allowed time saving.
- The extension, refinement, adaptation of the draft process meta-model is unlimited. Method engineers are free to metamodel the process that the organisations really need. Our method allows flexibility and adaptability.

6. Discussion

In this paper, we have presented a method to build process metamodels for information systems engineering, thus supplying the “rigour” element sought by Booch (1991) as noted earlier. In this section, we discuss some works related to the information systems engineering process in terms of modelling levels, coverage of the viewpoints, and thus flexibility and adaptability of the models.

6.1. Modelling levels

(Fiorini et al., 2001) present a Process Reuse Architecture. This architecture allows storing, classifying and retrieving process frameworks, process patterns or common processes in order to reuse them to build process models. Our solution is different because we provide a method to build process metamodels.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

(Tran et al., 2007) propose a similar approach to the Process Reuse Architecture. They provide process patterns to build or improve process models. To structure the process patterns, they defined a metamodel for process pattern, which is comparable to pattern formalism, a metamodel to represent the structural process model, a metamodel to represent the relations between the process elements and a metamodel to represent the behavioural process model. Then, they can reuse process patterns on existing process models, in order to improve or extend them, or they can build new process models. The result of this approach is different from ours in that method engineers work at the modelling level: they only modify process models without changing the process metamodels.

Our proposed SMME approach utilizes ideas from SME and indeed provides the metamodel that can underpin method fragments from which a situational process model (method) can be constructed. From original ideas proposed by e.g. Bergstra et al. (1985), Kumar and Welke (1992), Harmsen et al. (1994), Harmsen (1997) and Brinkkemper et al. (1999), the SME literature has grown rapidly, some of the more recent papers being presented at the IFIP Conference in 2007 (Ralyté et al., 2007).

Overall, our approach is limited by the number of application instances. In other words, as noted earlier, for the majority of cases, SME using a pre-defined standardized metamodel will suffice and provide the organization with benefits in both time and quality of the situational method that they construct and utilize. However, the use of metamodels is still in its infancy in software engineering, although having matured rapidly over the last ten years (as detailed, for instance, in Gonzalez-Perez and Henderson-Sellers, 2008). Metamodels are now seen as central to a number of emerging areas in software engineering, including Model-Driven Engineering (MDE) as well as to agent-oriented software engineering. As standardized metamodels themselves become appreciated, understood and used more, some organizations will soon recognize the constraints of such pre-determination much as over the last ten years developers have realized the problems of a one-size-fits-all methodology and opted for the construction of a situated method. Much as SME has emerged from methodological thinking, SMME will itself emerge from the same perceived limitations but at a higher level of abstraction – an increasing level of abstraction over time being a general trend in very many aspects of computing, including information systems engineering.

6.2. Coverage

In the Process Reuse Architecture (Fiorini et al., 2001) the definition of a process is restricted to our definition of activity-oriented process. In (Tran et al., 2007), their process metamodel does not include the product-oriented, decision-oriented, context-oriented and strategy-oriented concepts. Thus, the process models produced or worked are only activity-oriented.

Henderson-Sellers and Gonzalez-Perez (2005) present a process metamodel for software development methodologies and their enactment. This process metamodel comprises producers, work products, work units and stages. However, there are no explicit decision, context and strategy viewpoints.

Although the published literature is indecisive on the nature of the link between metamodels and ontologies, they are clearly closely affiliated (e.g. Gonzalez-Perez and Henderson-Sellers, 2006a) but not isomorphic. We therefore include in this discussion a recent and very interesting approach in which Leppänen (2007) presents an ontology for information systems 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), facilities, time and location. The context in ISD is the composition of the seven domains, which differs from (Rolland et al., 1995). However, there is no domain concerning strategy.

6.3. Flexibility

Working at the modelling level is much more restrictive than working at the metamodelling level. Therefore, approaches focusing on activity process models offer less flexibility and adaptability than our method, which allows the construction of process metamodels that include various viewpoints. The existing approaches can be appropriate only in the case when method engineers do not need to represent unusual concepts, for example, if they only focus on the activity oriented viewpoint of the process models. Nevertheless, once method engineers need to represent decisional concepts or intentional concepts linked with operational concepts in their process models, there is no existing method. Our SMME approach really satisfies the necessity for flexibility and adaptability according to the organisational and project context.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

7. Conclusion

In this paper, we present different existing process metamodels and their main concepts, distinguishing five categories of process metamodels called viewpoints: activity, product, decision, context and strategy. We expose the different problems linked with their use and their capability to fulfil the organisation's requirements.
For situations when standard metamodels are inadequate, we propose a method to build unified, multi-viewpoint and fitted process metamodels. The two-phase method described here uses a process domain metamodel and patterns. The first phase, selection, allows the method engineer to select the main concepts necessary from the process domain metamodel by using a questionnaire. The second phase, refinement, allows the method engineer to refine and enrich the draft process metamodel obtained in the first phase, reusing metamodelling patterns and business patterns. The method engineer then obtains a complete process metamodel fulfilling the organisational needs. This process metamodel can include all the viewpoints if needed, and the correspondence between their concepts. The method engineer can then instantiate the process metamodel to model the different required process models.
To improve the resources, we need to define which design patterns can be useful to metamodelling and then adapt them. The process domain metamodel can be improved by method engineer experts who are responsible for technology watch on methods, process models and process metamodels. If needed, they can add new main concepts to the process domain metamodel.
Finally, the method could be implemented as a workflow, based on the system pattern stored in the AGAP tool. Future work includes the development of a CAME (Computer-Aided Method Engineering) tool (Tolvanen, 1998) to model the process metamodel and instantiate it. The process model should be represented with the desired formalism. Industry evaluation will, in the future, supply more insights into the situations in which SMME as well as SME will be beneficial to the organisation.

Acknowledgment

We would like to express our special thanks to the Association Française des Femmes Diplômées des Universités (AFFDU) of Grenoble branch, for its financial support.
This is contribution number 08/02 of the Centre for Object Technology Applications and Research of the University of Technology, Sydney.

References

Alexander, C., 1979. The Timeless Way of Building, Oxford University Press.

Atkinson C., Kühne T., 2001. The Essence of Multilevel Metamodeling, UML'01, LNCS 2185, 19-33, Springer, Berlin Heidelberg. doi:10.1007/3-540-45441-1_3.

Australian Standard, 2004. Standard Metamodel for Software Development Methodologies, AS 4651—2004.

Beck, K., 1999. Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, Longman Publishing Co., Inc. Boston, Massachusetts.

Bergstra, J., Jonkers, H., Obbink, J., 1985. A Software Development Model for Method Engineering. In Roukens, J., Renuart, J., (Eds), Esprit'84: Status Report of Ongoing Work, 85-94, Elsevier Science Publishers, Amsterdam.

Booch, G., 1991. Object Oriented Analysis and Design with Application, Benjamin-Cummings Publishing Co., Inc. Redwood City, California.

Brinkkemper, S., 1996. Method engineering: engineering of information systems development methods and tools, Inf. Software Technol., 38(4), 275-280. doi:10.1016/0950-5849(95)01059-9.

Brinkkemper, S., Saeki, M., Harmsen, F., 1999. Meta-Modelling Based Assembly Techniques for Situational Method Engineering. Inf. Syst (24), 209-228 doi:10.1016/S0306-4379(99)00016-2.

Cauvet, C., 2006. Modélisation des processus d'ingénierie des Systèmes d'Information, Encyclopédie de l'Informatique et des Systèmes d'Information, 1412-1425, Vuibert, Paris.

Coad, P., 1992. Object-Oriented Patterns, Communication of the ACM, ACM Press 35(9), 152-159. doi:10.1145/130994.131006.

Conte, A., Fredj, M., Hassine, I., Giraudin, J.-P., Rieu, D., 2002. A Tool and a Formalism to Design and Apply Patterns, OOIS, LNCS 2425, 135-146, Springer, Berlin Heidelberg.

Dowson, M., 1987. Iteration in the software process, Review of the 3rd International Software Process Workshop, Proceedings of the 9th International Conference on Software Engineering, IEEE Computer Society Press, 36-41.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

Eclipse Process Framework Project, 2006. http://www.eclipse.org/epf.

Finkelstein, A., Kramer, J., Goedicke, M., 1990. ViewPoint oriented software development, third International Workshop on Software Engineering and Its Applications, 374-384.

Fiorini, S.T., Do Prado Leite, J.C.S., De Lucena, C.J.P., 2001. Process Reuse Architecture. CAiSE'01, LNCS 2068, 284-298, Springer, Berlin Heidelberg.

Forrest, J., 2008. Business Process Oriented Requirements Modelling and Systems Fulfilment - a Meta Model Driven Approach, ASWEC, Experience Report Proceedings, IEEE Computer Society Press, Los Alamitos, CA, USA, 88-97.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., 1995. Design patterns -Elements of reusable object-oriented software, Professional Computing Series, Addison Wesley, Longman Publishing Co., Inc. Boston, Massachusetts.

Gonzalez-Perez, C., 2005. Tools for an Extended Object Modelling Environment, ICECCS, 20-23. doi:10.1109/ICECCS.2005.80.

Gonzalez-Perez, C., Giorgini, P. Henderson-Sellers, B., 2008. Method construction by goal analysis. In Barry, C., Lang, M., Wojtkowski, W., Wojtkowski, G., Wrycza, S., and Zupancic, J. (Eds), The Inter-Networked World: ISD Theory, Practice, and Education, Springer-Verlag, New York.

Gonzalez-Perez, C., Henderson-Sellers, B., 2006a. An Ontology for Software Development Methodologies and Endeavours. In Calero, C., Ruiz F., Piattini, M. (Eds), Ontologies in Software Engineering and Software Technology, 123-152, Springer-Verlag, Berlin.

Gonzalez-Perez C., Henderson-Sellers B., 2006b. A powertype-based metamodelling framework, Software and System Modeling 5(1), 72-90. doi:10.1007/s10270-005-0099-9.

Gonzalez-Perez, C. Henderson-Sellers, B., 2007. Methodology enactment using a work product pool approach, J. Systems and Software. doi:10.1016/j.jss.2007.10.001.

Gonzalez-Perez, C. ,Henderson-Sellers, B., 2008. Metamodelling for Software Engineering, J. Wiley & Sons, Chichester.

Grosz, G., Rolland, C., 1990. Using Artificial Intelligence Techniques to Formalize the Information System Design Process, DEXA 1990, 374-380, Springer, Berlin Heidelberg.

Harel, D., 1987. Statecharts: A Visual Formulation for Complex Systems, Science of Computer Programming 8(3), 231-274.

Harmsen, A.F., 1997. Situational Method Engineering, Moret Ernst & Young.

Harmsen, A.F., Brinkkemper, S., Oei, H., 1994. Situational Method Engineering for Information Systems Projects. In Olle, T.W. , Verrijn-Stuart, A.A. (Eds), Methods and Associated Tools for the Information Systems Life Cycle, Proceedings of the IFIP WG8.1 Working Conference Cris/94, 169-194, IFIP Transactions, vol. A-55. Elsevier Science, New York.

Henderson-Sellers, B., Gonzalez-Perez, C., 2005. A comparison of four process metamodels and the creation of a new generic standard, Inf. Software Technol 47, 49-65. doi:10.1016/j.infsof.2004.06.001.

Henderson-Sellers, B. Gonzalez-Perez, C., 2006. On the ease of extending a powertype-based methodology metamodel, WoMM 2006, LNI P-96, 11-25.

Henderson-Sellers, B., Nguyen, V.P., 2004. Un outil d'aide à l'ingénierie de méthodes reposant sur l'approche OPEN, Génie Logiciel 70, 17-28.

Humphrey, W.S., Kellner, M. I. 1989. Software Process Modeling: Principles of Entity Process Models, ICSE 1989, IEEE Computer Society / ACM Press, 331-342. doi:10.1145/74587.74631.

Hug, C., Front, A., Rieu, D., 2007. Ingénierie des processus: une approche à base de patrons, INFORSID, 471-486.

Hug, C., Front, A., Rieu, D., 2008. A Process Engineering Method Based on Ontology and Patterns. ICSOFT (ISDM/ABF), 29-36.

ISO/IEC 24744, 2007. Software Engineering — Metamodel for Development Methodologies.

Jarke, M., Mylopoulos, J., Schmidt, J.W., Vassiliou, Y., 1992. DAIDA: An Environment for Evolving Information Systems, ACM Transactions on Information Systems 10(1), 1-50. doi:10.1145/128756.128757.

Karagiannis, D., Kühn, H., 2002. Metamodelling platforms, in E-Commerce and Web Technologies, LNCS 2455, 451-464, Springer-Verlag, Berlin.

Kruchten, P., 2000. The Rational Unified Process: An Introduction. Addison-Wesley, Longman Publishing Co., Inc. Boston, Massachusetts.

Kumar, K., Welke, R.J., 1992. Methodology Engineering: A Proposal for Situation-Specific Methodology Construction. In Cotterman, W.W., Senn, J.A. (Eds), Challenges and Strategies for Research in Systems Development, 257-269, John Wiley & Sons: Chichester, UK.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

Kunz, W., Rittel, H.W.J., 1970. Issues as elements of information systems. Working Paper 131, Heidelberg-Berkeley.

Leppänen, M. 2007. Towards an Ontology for Information Systems Development – A Contextual Approach. In: Siau K. (ed.) Contemporary Issues in Database Design and Information Systems Development, IGI Publishing, New York, 1-36.

Mi, P., Scacchi, W. 1996. A meta-model for formulating knowledge-based models of software development. Decis. Support Syst. (17), Elsevier Science Publishers B. V., 313-330 doi:10.1016/0167-9236(96)00007-3.

MP, Bonnes pratiques de métamodélisation et patrons pour la méta-modélisation, 2007. http://planet-mde.org/idm07/

Odell J., 1994. Power Types. JOOP 7(2), 8-12.

OMG, 2005. Software Process Engineering Metamodel Specification. Version 1.1.

OMG, 2007. Unified Modeling Language: Superstructure. Version 2.1.1.

OOSPICE, Software Process Improvement and Capability Determination for Object- Oriented/ Component-Based Software Development, 2002. www.oospice.com.

Open Process Framework, 2005. http://www.opfro.org.

Plihon, V., 1996. Un environnement pour l'ingénierie des méthodes. Ph.D. Thesis, University of Paris I, Paris, France.

Potts, C., 1989. A generic model for representing design methods, ICSE '89, ACM Press, 217-226. doi:10.1145/74587.74616.

Potts, C., Bruns, G., 1988. Recording the Reasons for Design Decisions, ICSE'88, IEEE Computer Society Press, 418-427.

Ralyté, J., Brinkkemper, S., Henderson-Sellers, B. (Eds), 2007. Situational Method Engineering: Fundamentals and Experiences. Proceedings of the IFIP WG 8.1 Working Conference, 12-14 September 2007, Geneva, Switzerland, IFIP Series, Vol. 244, Springer, Berlin.

Rolland, C., 1998. A comprehensive view of process engineering, CAiSE'98, LNCS 1413, 1-24, Springer, Berlin Heidelberg.

Rolland, C., Nurcan, S., Grosz, G., 2000. A decision-making pattern for guiding the enterprise knowledge development process, Inf. Software Technol. 42(5), 313-331. doi:10.1016/S0950-5849(99)00089-0.

Rolland, C., Prakash, N., Benjamen, A., 1999. A Multi-Model View of Process Modelling, Requirements Engineering 4(4), 169-187. doi:10.1007/s007660050018.

Rolland, C., Souveyet, C., Moreno, M., 1995. An Approach for defining ways-of-working, Information System Journal 20(4), 337-359. doi:10.1016/0306-4379(95)00018-Y.

Schwaber, K., Beedle, M., 2001. Agile Software Development with SCRUM, Prentice Hall, Upper Saddle River, New Jersey.

Tolvanen, J.-P., 1998. Incremental Method Engineering with Modeling Tools. Dissertation, Jyväskylä Studies in Computer Science, Economics and Statistics, Vol. 47, University of Jyväskylä, Finland.

Tran, H. N., Coulette, B., Dong, B. T., 2007. Modeling Process Patterns and Their Application, ICSEA'07, IEEE Computer Society, 15-20. doi:10.1109/ICSEA.2007.52.

Zoukar, I., Salinesi, C., 2004. Using goal/strategy/maps to reduce the language disparity issue in ERP projects. In Grundspenkis, J., Kirikova, M. (Eds.), Knowledge and Model Driven Information Systems Engineering for Networked Organisations, Proceedings, 325-339, Vol. 2. Faculty of Computer Science and Information Technology, Riga.

Zoukar, I., 2005. MIBE : Méthode d'Ingénierie des Besoins pour l'implantation d'un progiciel de gestion intégré (ERP). Ph.D. Thesis, University of Paris I, Paris, France.

Charlotte Hug is PhD Student in Information Systems at the Laboratory of Informatics of Grenoble (LIG) and Joseph Fourier University (Grenoble, France) where she took her master's research degree in 2006. Her research areas include information systems engineering, process metamodelling and development methods. She is also junior lecturer at Pierre Mendès France University.

Agnès Front is an assistant professor at Pierre Mendès France University since 1998. Her research interests concern reuse-based information systems engineering: patterns based approaches, components based development methods, business process and services based development methods, MDE approach. She is co-responsible of the working group MADSI (Méthodes Avancées de Développement de Systèmes d'Information) in theme 3 of GDR I3 and member of the executive committee of INFORSID association.

dx.doi.org/10.1016/j.jss.2009.05.020
Journal of Systems and Software - Volume 82, Issue 10, October 2009, Pages 1730–1742

Dominique Rieu is full professor at Pierre Mendès France University. She is joint director of Laboratory of Informatics of Grenoble (LIG) and vice-president Information System of Pierre Mendès France University. Her research interests concern information systems engineering (reuse, traceability and variability) and development methods (process modelling, process reuse). She is responsible of theme 3 of GDR I3 (engineering for and by models in information systems) and member of INFORSID executive committee.

Brian Henderson-Sellers is the Director of the Centre for Object Technology Applications and Research and Professor of Information Systems at the University of Technology, Sydney (UTS). He is author or editor of 27 books and is well-known for his work in object-oriented and agent-oriented methodologies (MOSES, COMMA, OPEN, OOSPICE, FAME), objectoriented metrics and metamodelling. More recently, he has chaired workshops on these topics at OOPSLA and AOIS (Agent-Oriented Information Systems). He is Editor of the International Journal of Agent-Oriented Software Engineering and on the editorial board of Journal of Object Technology, Software and Systems Modeling and International Journal of Cognitive Informatics and Natural Intelligence. In July 2001, Professor Henderson- Sellers was awarded a Doctor of Science (DSc) from the University of London for his research contributions in object-oriented methodologies.