HAL
open science

ProMISE: a Process Metamodelling Method for Information Systems Engineering

Charlotte Hug, Agnès Front, Dominique Rieu

To cite this version:

Charlotte Hug, Agnès Front, Dominique Rieu. ProMISE: a Process Metamodelling Method for Information Systems Engineering. Evaluation of Novel Approaches to Software Engineering 5th International Conference, ENASE 2010, Athens, Greece, July 22-24, 2010, Revised Selected Papers, Springer-Verlag, pp.89-105, 2011, Communications in Computer and Information Science, 10.1007/978-3-642-23391-3 . hal-00662721

HAL Id: hal-00662721

https://paris1.hal.science/hal-00662721

Submitted on 28 Jan 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

ProMISE, a Process Metamodelling Method for Information Systems Engineering

Charlotte Hug[1], Agnès Front[2], Dominique Rieu[2],

[1] Centre de Recherche en Informatique, Paris 1 University, 90 rue de Tolbiac
75013 Paris, France
[2] Laboratoire d'Informatique de Grenoble, Grenoble University, 220 rue de la Chimie,
38041 Grenoble cedex 9, France
Charlotte.Hug@univ-paris1.fr, {Agnes.Front, Dominique.Rieu}@imag.fr

Abstract. Processes play a great part in information systems engineering projects success. There are a lot of process models and metamodels; however, the "one size fits all" motto has to be moderated: models have to be adapted to the specificities of the organizations or the projects. In order to help method engineers building adapted process models, we propose a method to build process metamodels and to instantiate them according to the organizations context. Our method consists of selecting the concepts needed from a conceptual graph, gathering the current knowledge of metamodelling concepts for information systems engineering processes, and integrating them in a new process metamodel that will be instantiated for any project in an organization. This method is supported by a tool.

Keywords: Process engineering, information systems engineering, metamodelling, graph, tool.

1 Introduction

To design and produce information systems, project managers focus on the quality of the deliverables or on the intermediary support documents produced all along the project life (analysis models, test procedures, for example); as such, they focus on the quality of their definition, formalization, level of detail and completeness. The quality of the products highly depends on the processes followed [1], as the processes define the way products have to be created. A development process can be roughly defined as a sequence of activities that create and update products. The objective for an organization is to properly define the processes, formalize them, adjust them to the different projects and reproduce the optimized processes. The Capability Maturity Model Integration [2] specifies different degrees of maturity of the development processes in an organization, the supreme goal being following repeatable and optimized processes. The information systems engineering (ISE) processes quality is then essential.

Many information systems/software engineering processes or methods have been defined. They appeared in the 1970's with the Waterfall model [3], the Spiral Model

[4], then the RUP [5] and more recently Agile methods as XP [6] and SCRUM [7]. They are based on different process models: they propose different lifecycles and activities, specify distinct kinds of deliverables and assign roles differently. Thus, each method proposes its own way to build IS: each method is based on a different process metamodel that uses different concepts.

In order to produce information systems, process models have to be efficient and fitted to the organizations specific constraints. An unsuitable method or process model will not be followed by the development teams, create tensions between team members and generate delay or bad IS design. Existing methods or process models have then to be adapted, customized to the organizations context; this is the method engineer's role.

As the process models flexibility depends on their process metamodel flexibility, we state that the key to build adapted process models lies in adapted process metamodels. However, existing process metamodels are hardly adaptable and are defined independently of one another [8], [9], [10]. Upon modelling the process models of their organizations, method engineers have to use those already predefined process models or to instantiate process metamodels without adaptation possibilities; the resulting models might be partially inadequate to the organizations specificities and constraints and to their business activities.

In this paper, we present the ProMISE method (Process Metamodelling for Information Systems Engineering) that allows method engineers building their own process metamodels according to their organization specificities and technologies. The method consists of selecting the needed concepts from a conceptual graph and integrating them in a new adapted process metamodel. The construction of the process metamodel is hidden to the method engineers: they use a conceptual graph that builds the process metamodel and checks its consistency. The produce process metamodels are multi-points of view as they integrate various points of view of the existing process metamodels, they are adapted to the constraints and specificities of the organization as only the needed concepts are integrated and the process metamodel is federated as all the knowledge of ISE processes is defined in one metamodel.

The paper is organized as follows. In the next section, we present the conceptual graph, base of our adaptive method to build process metamodels for ISE. We introduce the method in Section 3. Section 4 presents an example of the Grenoble's University Hospital. Section 5 is devoted to discussion and Section 6 presents the tool that supports our method. Section 7 concludes this paper.

2 The base of the method: the Conceptual Graph

In this section, we present the base of our approach that is a conceptual graph. It was built from a Process Domain Metamodel and a 3D Space [8], [9], [10]. A study [10], [11] of the different existing process metamodels (activity oriented [12]; [13]; [14]; [15]; [16]) such as SPEM, product oriented [17]; [18];[19]; [1] such as Statechart and State Machines, decision oriented [20]; [21]; [22]; [23] like Ibis and Daida, context oriented [24] such as NATURE and strategy oriented [25] like MAP), allowed us to define a Process Domain Metamodel which only contains the main classes of existing

process metamodels and the associations between the concepts. In order to facilitate the classes' selection from the Process Domain Metamodel, we propose the use of a conceptual graph that allows method engineers to easily navigate between the concepts. The concepts are organized according to a 3D space.

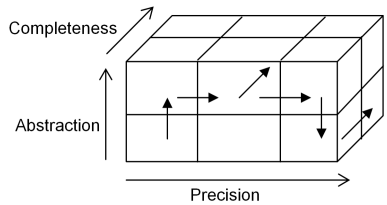

Fig. 1. The Completeness – Precision – Abstraction 3D space.

2.1 The 3D Space

The 3D space represented in Figure 1 guides method engineers through a methodological frame to build process metamodels for ISE. The three axes [26] help method engineers in the selection of the concepts: completeness, precision and abstraction. Completeness is the coverage of the metamodel of one or more points of view (activity, product, decision, context and strategy). Precision is the level of detail of the metamodel and abstraction is the intentional and/or operational level of concern of the metamodel. The intentional level represents the objectives of the ISE process while the operational level represents the actions required to concretize these objectives. Method engineers will build their process metamodels depending on these three axes: each engineering activity has for objective to: extend the Process Metamodel Under Construction (PMUC) (completeness axis), precise the PMUC (precision axis) or abstract (inv. concretize) the PMUC (abstraction axis).

2.2 The Conceptual Graph

The conceptual graph (Figure 2) is the base of our method. It organizes the recognized concepts for ISE process metamodelling, representing the actual knowledge base of the domain. The purpose of such conceptual graph is to guide method engineers in the Completeness – Precision – Abstraction 3D space while selecting the concepts they need to represent in their metamodels. The conceptual graph defines the set of possibilities: it restrains method engineers in the selection and the use of the defined concepts only, in order to maintain the consistency of the PMUC.

2.2.1 The Concepts

The concepts of the conceptual graph are used in ISE processes and are usually represented in process metamodels. The concepts of the graph represent two types of elements:

- Classes that represent the main concepts (concepts in bold in Figure 2) defined in the Process Domain Metamodel and are linked to each other by the completeness and abstraction relations. Those concepts are Work Unit, Condition and Role (activity point of view) [12]; [13]; [14]; [15]; [16], Work Product (product point of view) [17]; [18]; [19];[1], Issue, Alternative, Argument (decision point of view) [20]; [21]; [22]; [23], Situation, Context, Intention (context point of view) [24] and Strategy (strategy point of view) [25]. Figure 3 presents a close-up on a few of those. A Work Unit represents an action that is executed during the ISE process. A Work Product is something that is produced, used or modified during the ISE process and a Role is someone/thing that carries out an action during the ISE process. A Strategy represents how an intention is achieved.

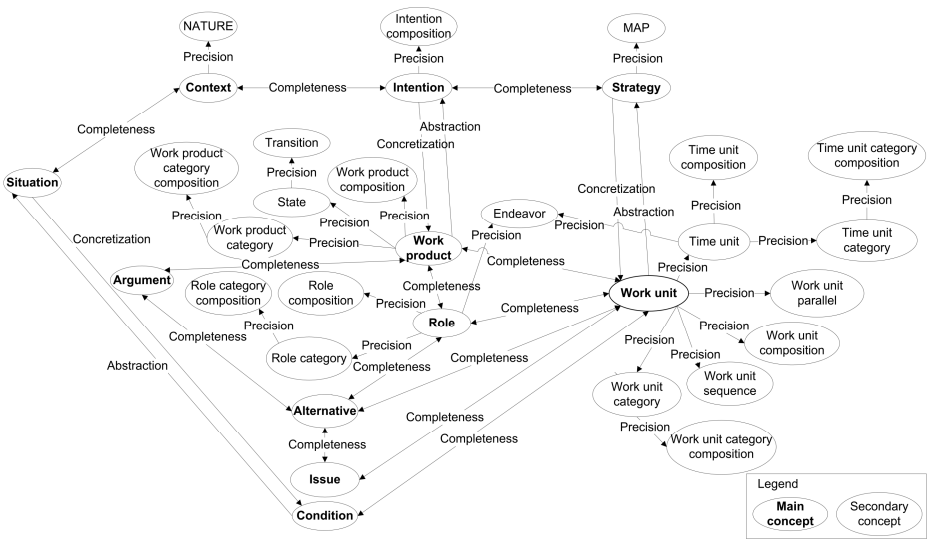

Fig. 2: The conceptual graph.

- Classes that decompose the previous classes, linked by the precision relation (secondary concepts). For example, in Figure 3, the Work Unit Category concept refines the Work Unit concept to express the fact that there are different categories of work unit, as activity or task for example. The Work Unit Composition concept refines the Work Unit concept to represent a Work Unit class with a reflexive composition, to express that the "Design components" activity is composed of the tasks "Class design" and "Subsystem design" [5], for example.

2.2.2 The Relations

The relations represent conceptual links between concepts in the Completeness – Precision – Abstraction 3D space as presented in section 2.1.

The completeness relation links one concept to another that extends it. This relation is symmetric, non-transitive and non-reflexive. For example, in Figure 3 (on the left), the Work Unit concept can be completed by the Work Product and Role

concepts. As the Work Product concept can also be completed by the Work Unit concept (symmetry), the represented link is bidirectional.

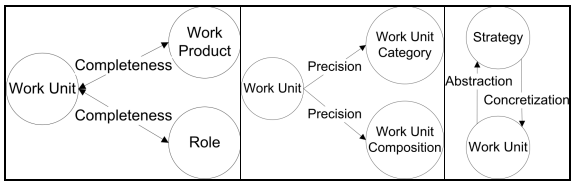

Fig. 3: Examples of the Completeness, Precision and Abstraction relations.

The precision relation specifies that a concept can be refined by another concept. Such relation is non-symmetric, non-reflexive and non-transitive. For example, the Work Unit concept can be refined using the Work Unit Category or Work Unit Composition concepts (but the Work Unit concept does not refine the Work Unit Category concept – non symmetry) (cf. Figure 3 in the centre).

The abstraction relation specifies that one concept can be abstracted by another concept; it is non-symmetric, non-reflexive and non-transitive. For example, the Work Unit concept is abstracted by the Strategy concept (cf. Figure 3 on the right). The inverse relation of Abstraction is Concretization. We can say that the Work Unit concept is the concretization of the Strategy concept.

On the one hand, the relations help method engineers selecting the concepts in the conceptual graph and on the other hand, they assure the coherency of the selected concepts. For example, the Work Unit Category Composition concept can not be selected before the Work Unit Category concept (Figure 2). The consistency of the process metamodels produced is then ensured, as the conceptual graph was designed in such a way as the concepts were coherently linked to each others.

2.2.3 Example

The conceptual graph in the Completeness – Precision – Abstraction 3D space is dynamically built: the perspective evolves depending on the node the method engineer is considering. Figure 4 shows a part of the 3D perspective that method engineers would see from the Work Unit concept.

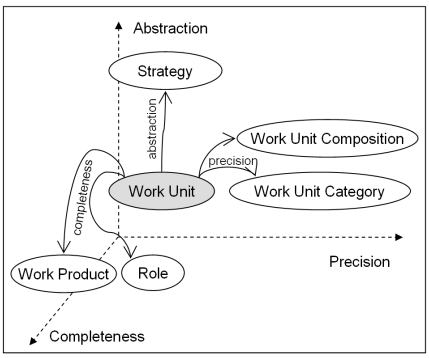

Fig. 4: Part of the perspective from the Work Unit concept in the conceptual graph.

If method engineers want to extend their PMUC, it will lead to the Work Product and Role concepts thanks to the completeness relation defined in the conceptual graph. If they want to precise their PMUC, it will lead to the Work Unit Category and Work Unit Composition concepts, using the precision relation and if they want to abstract it, it will lead to the Strategy concept thanks to the abstraction relation.

We now describe the method that uses the conceptual graph to build process metamodels for ISE.

3 The method

In this section, we present the method based on the conceptual graph to build process metamodels for ISE. The two-step method consists of: (i) concepts selection within the conceptual graph, (ii) concepts integration in the PMUC, according to the Process Domain Metamodel. These two steps are iterated until method engineers obtain the complete process metamodel they need.

3.1 Concept selection

The first action of the Concept selection activity is the *Definition selection* that will lead to get a *Concept* (left part of Figure 5). A definition is composed of a short description, synonyms of the concept and examples (Table 1). It enables method engineers to select definitions from the *Concepts dictionary* corresponding to their needs. Each definition is associated to a concept appearing as a node in the conceptual graph. The next step is the Concept integration (section 3.2).

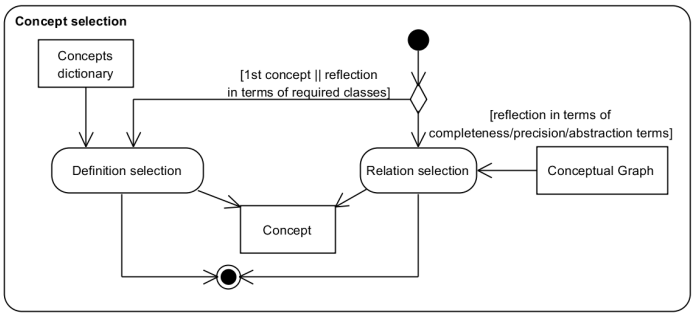

Fig. 5: The Concept selection.

After the first loop, method engineers go back to the *Concept selection*. They may refine the PMUC in terms of concepts attainable through relations with the previously integrated concept (completeness, precision and abstraction relations) or in terms of integration of classes thanks to the definitions. The *Relation selection* activity consists of selecting one of the relations that starts from the integrated concepts. For example, if the method engineer just integrated the Work Unit concept to his/her PMUC and if he/she wants to extend it, he/she could select Role, Work product and all the concepts

linked through the completeness relation to the Work Unit concept in the conceptual graph. It works in the same way through the precision and abstraction relations.

Table 1: Some definitions examples.

Description	Synonyms, AKA, examples	Concept
Represents how an intention is achieved	Tactics, approach, manner	Strategy
Objective of the ISE process	Goal	Intention
Task that is executed during the ISE process	Activity, task, work definition	Work Unit
Work Unit that is composed of other work units	Activity composed of tasks	Work Unit composition
Something that is produced, used or modified by a work unit during the ISE process	Product, document, model, program	Work Product
Someone/thing that carries out a work unit during the ISE process	Actor, developer, analyst, system	Role

3.2 Concept Integration

Once the concept is selected, it has to be integrated in the *PMUC*. The integration activity is rather complex (Figure 6): it has to take into account the different types of concepts (main or secondary).

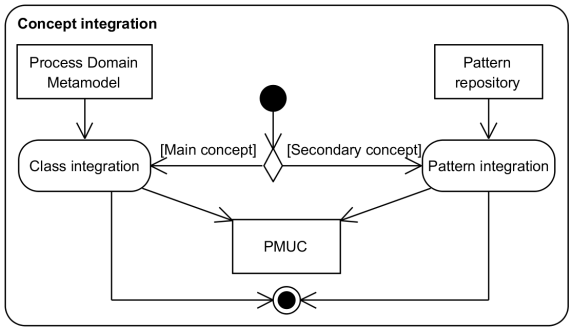

Fig. 6: The Concept integration.

The main concepts of the Conceptual Graph correspond to classes in the Process Domain Metamodel. These classes have then to be integrated in the PMUC with the associations between the integrated classes. The secondary concepts correspond to design or business patterns that are applied on the classes of the PMUC. The patterns that can be used are stored in a *Pattern Repository*. According to the selected concept, one of the patterns is applied on the PMUC. The PMUC is thus built by adding classes and applying patterns. The integration process is fully described in [27].

Method engineers can then choose either to continue the process or to stop it if the PMUC is complete. If the PMUC is not complete, they go back to the *Concept selection* activity.

The ProMISE method allows method engineers to build process metamodels according to the constraints and specificities of their organization as they only select the needed concepts from the conceptual graph. The conceptual graph allows guiding method engineers in the construction and checking the consistency of their PMUC. The guiding is done thanks to the relations defined between the concepts that method engineers will select according to their intention (abstract, complete, precise a concept). The consistency of the produced PMUC is continuously checked as method engineers can only select concepts according to the conceptual graph which have been built in order to verify the consistency at any time. Some concepts cannot be selected until other concepts have been integrated. Moreover, the construction of the process metamodel itself is hidden to the method engineers as they only manipulate the conceptual graph and the concepts definition.

We will now present an example of the ProMISE method use.

4 Grenoble's University Hospital Example

This section describes an example of the information system centre of Grenoble's University Hospital (http://www.chu-grenoble.fr/). This example has not a purpose of validating our method but illustrating it. We specifically conducted qualitative evaluations to validate the method with an academic focus group and semi-structured interviews with industrialists [28].

4.1 Requirements

The information system centre (ISC) manages approximately forty different applications that need to be regularly updated to meet new users' requirements (medical assistants, hospital doctors and administration staff).

The ISC managers want to model the ISE processes to achieve a more rigorous project management, defining a unified and optimal way to manage projects regardless of the development team. They also want to collect and reuse knowledge for a more efficient production in terms of resources and time use and therefore costs. A method engineer is in charge of the study of the ISE processes and their modeling. The method engineer in this example is one of the project managers of the ISC.

We have worked with this project manager who determined the various aspects of the ISE processes (this example only presents an extract of the problem):

- A part of the process is defined in terms of goals and sub-goals; this part is intended primarily for hospital services managers (services are for example the surgical unit or the accounting department) who are more interested in the results and impacts of new system functionalities on their service (intentional part),
- The second part of the process is defined by phases, activities and products produced during these activities (operational part).

The problems met by the method engineer are the following: how can he represent these concepts? What are the existing models? Which models meet these requirements? At the present time, these representation choices are made difficult because of the numerous existing process models and metamodels, their lack of

mutual complementarity and the complexity to adapt them to specific needs of organizations.

Our method enables the method engineer to model the process metamodel that corresponds to the information system centre ISE processes. The method guides him through the selection of concepts he needs to represent and through their assembly in order to create a specific process metamodel including all the concepts at the intentional level concerning the services managers and at the operational level concerning the activities and the products.

4.2 Method Use

The first step of our method is the Concept selection. The method engineer must select one of the definitions that correspond to the concepts he wants to model. The definition "Goal or objective of the ISE process" corresponds to the part of the process defined in terms of goals. The engineer chooses this definition and the corresponding Intention class from the Process Domain Metamodel is integrated in the new PMUC. The method engineer examines then the relations of the Intention concept in the conceptual graph; the *precision* relation permits him to select the Intention Composition concept that will allow him to decompose the goals into sub-goals. This concept is integrated in the PMUC as a reflexive composition on the Intention class, which corresponds to the use of the Composition pattern on the Intention class. Figure 7 presents this part of the path in the conceptual graph and the corresponding PMUC.

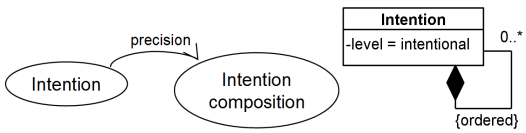

Fig. 7: First part of the path in the conceptual graph and the PMUC.

Then, the relation *concretization* starting from the Intention concept in the conceptual graph allows the method engineer to get the Work Product concept that will represent the products produced during the ISE process. The corresponding class is integrated in the PMUC, as well as the "concretizes" dependency linked to the Intention class. In order to model the fact that a work product can be composed of other work products (for example, "Functional specifications" is composed of "Simplified requirements" and "Actors diagram"), the method engineer refines the Work Product concept thanks to the Work Product Composition concept. To specify that work products are of different types (for example, "Functional specifications" is a document and "Actor diagram" is a UML diagram), the method engineer refines the Work Product concept by the Work Product Category concept. The Work Product Category class is added into the PMUC. Similarly to what was done with the Work Product, the method engineer wants to specify that a document is composed of UML diagrams, texts and graphics. He refines the Work Product Category concept by the Work Product Category Composition concept. Figure 8 presents the corresponding part of the path in the conceptual graph and the corresponding PMUC.

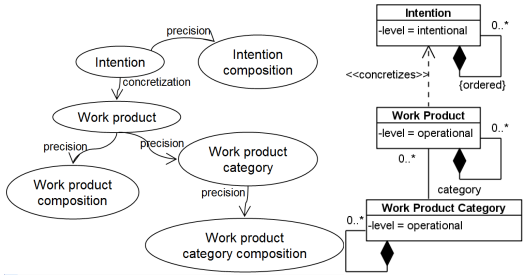

Fig. 8: Second part of the path in the conceptual graph and the PMUC.

Thanks to the *completeness* relation, the method engineer can extend the PMUC with the Work Unit concept to represent activities and steps. The Work Unit class and its associations "In" and "Out" defined in the Process Domain Metamodel are integrated to the PMUC. By using the *precision* relation, the method engineer can refine the Work Unit concept to represent the sequence and the composition of work units, the work unit categories and the composition of work unit categories. Figure 9 presents the complete path carried out in the conceptual graph.

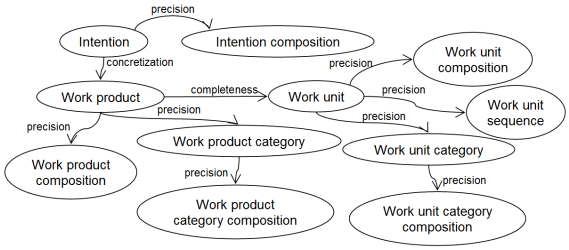

Fig. 9: Complete path in the conceptual graph.

Figure 10 presents the final process metamodel obtained. It represents the classes defined in the requirements and the associations between them. The link between the classes of intentional and operational level is represented by the dependency link stereotyped as "concretizes". The abstraction level of each class is represented as an attribute *level*. The process metamodel is multi-points of view as it focuses on the activity, product and strategy points of view. The complementarity and the connection between the points of view are modeled by the "concretizes" dependency and the associations.

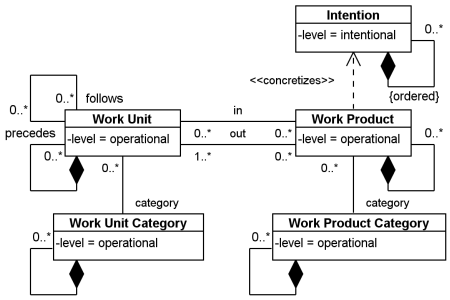

Fig. 10: The final process metamodel.

The method engineer can then instantiate the metamodel to represent the various ISE process models of the ISC. Figure 11 is a partial instantiation of the final process metamodel to represent the ISE processes. The method engineer wants to model the intentions and sub-intentions of service managers. One of the intentions of the service managers is to know the level of impact of a new functionality and the changes on the services organization. This can be represented as the object "Define the level of impact of the change in the service", instance of the Intention class. This intention can be decomposed into two sub-intentions. Service managers want to define the impact of the change in the service organization and the persons that will be impacted by the change. These estimations will be useful to define the costs of the IS change, as costs of business process modifications.

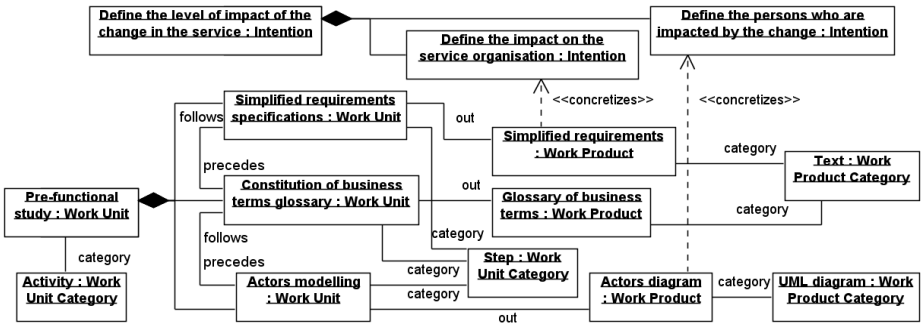

Fig. 11: The process model represented as an object diagram.

The operational abstraction level of the process model represents the detail of the "Pre-functional study" activity composed of three steps. First, "Simplified requirements specifications" produces the "Simplified requirements" work product that is a text. Second, the "Constitution of business terms glossary" step produces a glossary and finally, "Actors modeling" produces a UML diagram "Actors diagram". All the work products produced during the Pre-functional study form a document called "Functional specifications" (not represented in Figure 11). The two sub-intentions "Define the impact on the service organization" and "Define the persons

who are impacted by the change" are concretized by the "Simplified requirements" and "Actors Diagram" work products.

The process model represented as an object diagram is not easily and quickly understandable. Our method proposes a graphical representation (formalism) depending on the concepts in the PMUC. For example, if concepts of the operational level as work unit and work product are defined in the metamodel, the method will propose to use activity diagrams [18]. If intentions and strategies are used, the method will propose the MAP formalism [25], if there are only intentions, the KAOS formalism [29] will be proposed.

The top part of Figure 12 shows how the intentions and sub-intentions of the intentional level defined in Figure 11 can be modeled using the KAOS formalism. They are represented as parallelograms. The composition is modeled thanks to a circle. Figure 12 also presents the concepts of the operational level defined in Figure 11 as an activity diagram. The activities and steps are represented with rounded rectangles. All the work products are represented by rectangles. Stereotypes are used to specify their category. The "concretizes" dependencies are defined between the different work products and intentions of the models: the method engineer, the service managers and project managers can switch from the intentional level to the operational level.

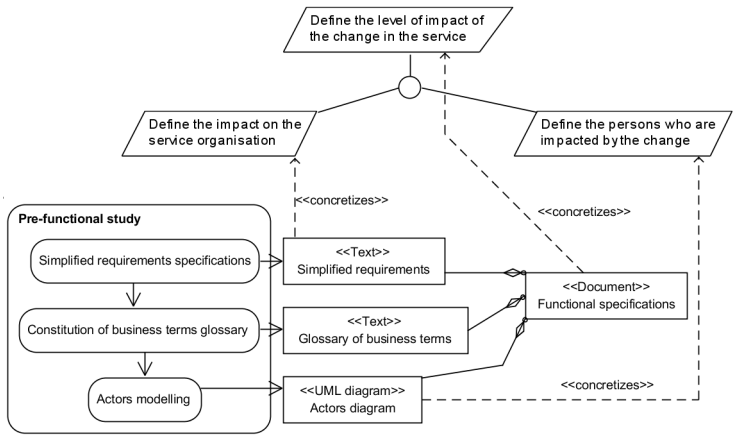

Fig. 12: Intentions and sub-intentions defined at the intentional level and their concretization at the operational level in the ISE process.

5 Discussion

Our proposition offers method engineers to build process metamodels for ISE depending on the specificities, the context of the projects or organizations. Our purpose differs from Situational Method Engineering, as its aim is to define IS development methods by reusing and assembling different existing method fragments

[30], but it is set in the same trend of situational engineering. We may name our domain SPME (Situational Process Metamodelling Engineering).

Let us note that we do not reconsider the existing process metamodels. They all play a part in ISE processes and have their legitimacy. However, they do not define their concepts complementarity in respect to the other process metamodels. Our proposition does not consist of yet another process metamodel, but it proposes a method allowing method engineers to build process metamodels including complementarity between the concepts. Our method uses some part of the existing process metamodels. Therefore, method engineers can reuse knowledge they acquired from their experience in ISE process metamodelling. There lies the real contrast between our proposal and currently available process models, such as RUP [5] or SCRUM [7], process models that are hardly adaptable. Applying these, method engineers must follow them as described and have a little or no mean of customization. Our method, on the other hand, proposes method engineers to instantiate process models according to their needs from process metamodels they have defined themselves but still using widely accepted concepts and formalism of ISE process models.

The existing process metamodels are also fixed [10]. They do not allow method engineers to extend them or customize them. Their use is therefore limited as they do not provide all needed concepts. For example, adding the intention concept to the RUP model would be difficult as it is not defined in the RUP metamodel. Using it without defining it in the metamodel could lead to misuses and the relations with the other concepts would not be defined.

Finally, new process metamodels as ISO/IEC 24744 [16] are more flexible and provide more concepts than previous process metamodels thanks to metamodelling mechanisms as the Powertype. However, the strategy, intention and decision concepts are not taken into account here.

To conclude, we can say that our method allows more flexibility, more personalized adaptation and allows building process metamodels with less limitation than the existing one.

6 The ProMISE tool

In this section, we present the ProMISE tool that supports our method. It has been built using Java. The two main supports of the method, the conceptual graph and the Process Domain Metamodel are defined independently from the tool in XMI files. XMI [31] is a standard format that allows storing UML models as structured text files. The main benefit of having the supports outside the tool is to permit more flexibility and scalability as the guiding will be generated thanks to the conceptual graph file and not the tool it-self. The guiding evolves as the conceptual graph evolves. Method engineers can interact with a visual conceptual graph, thanks to Prefuse [32]. Prefuse is a powerful toolkit for creating rich interactive data visualizations, such as graphs. The PMUC is displayed as a class diagram using the API UMLJGraph [33] that allows displaying UML diagrams in Java. The PMUC can be exported as an XMI file. This allows method engineers importing their process metamodels in any CASE tool,

to instantiate them for example. The imports and exports are done thanks to JDom [34], a Java API able to read and write both XML and XMI files.

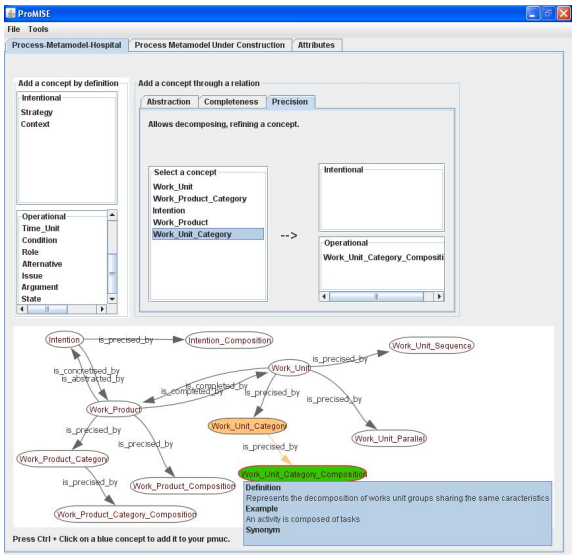

Fig. 13: Interface of the ProMISE tool.

The tool allows method engineers to build process metamodels through the use of the concepts definition and the relations. Figure 13 presents a global view of the interface. It is composed of three tabs:

– The first tab (here called “Process-Metamodel-Hospital) allows method engineers to build their PMUC for a particular organization or project through the use of the definitions and the conceptual graph.

– The second tab, “Process Metamodel Under Construction”, allows method engineers to view their PMUC as a UML class diagram.

– The third tab, “Attributes”, allows method engineers to add attributes to their PMUC classes, we will not detail this functionality here.

The first tab that allows the construction of the PMUC is decomposed in two parts:

– The top part of the interface permits to select concepts by definition or by relation. Concepts are displayed according to their abstraction level which facilitates their selection. The definition, examples and synonyms of each concept can be seen by mouse over. Each relation (completeness, precision, abstraction) is represented by a tab. By selecting one tab, the concepts that can be integrated through the corresponding relation are displayed in the lists. For example, in Figure 13, the Precision tab is selected. Work Unit Category is a concept that can be refined; this allows selecting the Work Unit Category Composition concept.

– The lower part of the interface shows the conceptual graph with the already integrated concepts in the PMUC and the concepts that can be reached by the relations and that can be integrated in the PMUC (Work Unit Category Composition in Figure 13). By selecting a relation tab, the conceptual graph is updated with the concepts that can be integrated.

The construction of the process metamodel itself is done by the tool that uses the Process Domain Metamodel, the patterns to add new classes to the PMUC. Method engineers do not see the “dirty” part of the process metamodel construction and only interact with the conceptual graph.

7 Conclusion

In this paper, we present a method that allows method engineers to build process metamodels for ISE. The method is based on two steps: (i) the selection of concepts meeting the specificities and constraints of the projects or organizations, using a conceptual graph to help the concepts selection in a completeness – precision – abstraction 3D space; (ii) the integration of the concepts to build an adapted process metamodel called PMUC. The produced process metamodels are multi-points of view as they integrate different points of view (activity, product, decision, context and strategy). The metamodels are also adapted to the context of the organizations as only the needed concepts were selected. At last, all the knowledge of ISE processes of the project or the organization is modeled in only one process metamodel and related process models. There is a better consistency of the manipulated concepts and a better understanding of the links between intentional and operational levels in the projects.

The ProMISE tool has been implemented to allow method engineers building process metamodels according to our method. The construction of the process metamodel itself is hidden to the method engineers as they only “play” with the conceptual graph: the process metamodel is built automatically by the tool.

Further step is to allow the instantiation of the process metamodels until the monitoring of particular information systems engineering projects. Another part of perspectives concerns the formalism that method engineers should use to represent the process models instantiated from the metamodels produced by this method. It would be useful to guide method engineers in the use of such or such formalism, depending on the concepts selected in their PMUC.

The Process Domain Metamodel may evolve, with the publications by the community of new process models and metamodels for ISE. The conceptual graph will also evolve, in order to propose method engineers the largest choice of possibilities taking into account the latest evolutions in terms of ISE process metamodelling.

References

1 Humphrey, W. S., Kellner, M. I.: Software process modeling: principles of entity process models. ICSE’89, pp 331--342. ACM, New York (1989)

2 Software Engineering Institute: CMMI for Development, Version 1.2 (2006)

3 Royce, W. W.: Managing the development of large software systems: concepts and techniques. ICSE’87, pp. 328--338. IEEE Computer Society Press (1987)

4 Boehm, B.: A spiral model of software development and enhancement. SIGSOFT Software Engineering Notes, vol. 11, n°4, pp. 14--24 (1986)

5 Kruchten, P.: The Rational Unified Process: An Introduction. Addison-Wesley, Longman Publishing, Co., Inc. Boston (2000)
6 Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley Professional, Longman Publishing Co., Inc. Boston (1999)
7 Schwaber, K., Beedle, M.: Agile Software Development with SCRUM. Prentice Hall, Upper Saddle River (2001)
8 Hug, C., Front, A., Rieu, D.: A Process Engineering Method Based on a Process domain Model and Patterns. MoDISE International Workshop, pp 126--137 (2008)
9 Hug, C., Front, A., Rieu, D.: Process Engineering Method Based on Ontology and Patterns. ICSOFT'08, pp. 29--36 (2008)
10 Hug, C., Front, A., Rieu, D., Henderson-Sellers, B.: A Method to build Information Systems Engineering Process Metamodels. J. of Sys. & Soft., vol. 82, n°10, pp. 1730--1742 (2009)
11 Hug, C., Front, A., Rieu, D.: Ingénierie des processus. Une approche à base de patrons. Revue RSTI. Série ISI, vol. 13, n°4, pp. 11--34. Hermès, France (2008)
12 OMG: Software Process Engineering Meta-Model. Version 2.0 (2008)
13 Open Process Framework, http://www.opfro.org
14 OOSPICE, Software Process Improvement and Capability Determination for Object-Oriented/ Component-Based Software Development, http://www.oospice.com
15 Australian Standard: Standard Metamodel for Software Development Methodologies. AS 4651—2004 (2004)
16 ISO/IEC: 24744 Software Engineering - Metamodel for Development Methodologies (2007)
17 Harel, D.: Statecharts: A Visual Formulation for Complex Systems. Science of Computer Programming, vol. 8, n°3, pp. 231--274 (1987)
18 OMG: Unified Modeling Language: Superstructure. Version 2.2 (2009)
19 Finkelstein, A., Kramer, J., Goedicke, M.: ViewPoint oriented software development. Third International Workshop on Software Engineering and Its Applications, pp. 374--384 (1990)
20 Kunz, W., Rittel, H.W.J.: Issues as elements of information systems. WP 131, Heidelberg-Berkeley (1970)
21 Potts, C., Bruns, G.: Recording the Reasons for Design Decisions. ICSE'88, IEEE Computer Society Press, pp. 418--427 (1988)
22 Potts, C.: A generic model for representing design methods. ICSE'89, pp. 217--226. IEEE Computer Society/ ACM Press (1989)
23 Jarke, M., Mylopoulos, J., Schmidt, J.W., Vassiliou, Y.: DAIDA: An Environment for Evolving Information Systems. ACM Trans. on Inf. Sys. vol. 10, n°1, pp. 1--50 (1992)
24 Rolland, C., Souveyet, C., Moreno, M.: An Approach for defining ways-of-working. Information System Journal, vol. 20, n°4, pp. 337--359 (1995)
25 Rolland, C., Prakash, N., Benjamen, A.: A Multi-Model View of Process Modelling. Requirements Engineering, vol. 4, n°4, pp. 169--187 (1999)
26 Panet, G., Letouche, R.: Merise/2 Modèles et techniques Merise Avancés. Les Editions d'Organisation, Paris (1994)
27 Hug, C.: Méthode, modèles et outil pour la méta-modélisation des processus d'ingénierie de systèmes d'information. PhD Thesis, Grenoble I University, (2009)
28 Hug, C., Mandran, N., Front, A., Rieu, D.: Qualitative Evaluation of a Method for Information Systems Engineering Processes. RCIS'2010, pp 257--268 (2010)
29 Objectiver: A KAOS tutorial. Respect-It (2007)
30 Ralyté J., Rolland, C.: An Assembly Process Model for Method Engineering. CAiSE 2001, LNCS, vol. 2068, pp. 267--283. Springer-Verlag, London (2001)
31 OMG.: MOF 2.0 / XMI Mapping Specification. Version 2.1.1 (2007)
32 Prefuse, http://prefuse.org/
33 UMLJGraph, http://umljgraph.sourceforge.net/
34 JDOM, http://www.jdom.org/