EPM–RT–2010-14

INTRODUCING KNOWLEDGE CONCEPTS IN SOFTWARE PROCESS MODELING

Noureddine Kerzazi, Pierre N. Robillard
Département de Génie informatique et génie logiciel
École Polytechnique de Montréal

Novembre 2010

EPM-RT-2010-14

INTRODUCING KNOWLEDGE CONCEPTS IN SOFTWARE PROCESS MODELING

Noureddine Kerzazi, Pierre N. Robillard
Département de génie informatique et génie logiciel
École Polytechnique de Montréal

Novembre 2010

EPM-RT-2010-14
Introducing Knowledge Concepts in Software Process Modeling
par : Noureddine Kerzazi, Pierre N. Robillard
Département de génie informatique et génie logiciel
École Polytechnique de Montréal

Introducing Knowledge Concepts in Software Process Modeling

Noureddine Kerzazi
Department of Computer and Software Engineering
École Polytechnique de Montréal
Montréal, Canada
+1 514 340 4711 # 7182
Noureddine.Kerzazi@polymtl.ca

Pierre N. Robillard
Department of Computer and Software Engineering
École Polytechnique de Montréal
Montréal, Canada
+1 514 340 4711 # 4238
Pierre-n.Robillard@polymtl.ca

ABSTRACT

Software process is knowledge intensive. Nevertheless, knowledge concepts are rarely taken into account in software process modeling. This paper presents a new software process modeling approach, which takes into account the various conceptual knowledge required to perform a task. The approach is based on the Software & Systems Process Engineering Meta-model (SPEM 2.0). It essentially adds knowledge attributes to existing relationships between roles, tasks and artifacts. Comparison between attributes for a given task provides information on the knowledge-gap between the SPEM elements involved. This information could be used in knowledge oriented project management to evaluate the risk associated to the knowledge gaps. A software tool has been implemented to facilitate the recording of various knowledge concepts while modeling the software process. Example of this approach is presented.

Keywords

Software Process Modeling; Software & Systems Process Engineering Meta-model (SPEM); Conceptual knowledge.

1. INTRODUCTION

Over the last 10 years there has been a greater interest around Knowledge creation and management for software organizations. Many authors claim, for different reasons, the integration of knowledge management (KM) in software development process to be knowledge-oriented. Dakhli and Ben Chouikha [1] think that software artifacts are accumulation of knowledge owned by organizational stakeholders. Rus and Lindvall [2] argued that KM is a risk prevention and mitigation strategy within software organizations. Moreover, the authors advocate the relevance of learning process, described as a fundamental part of KM, which support employees to perform specific tasks. Wang et al. [3] recall the fact that software processes are people-dependent within a context of creative work. Robillard [4] pointed out that for providing relevant support to software development, a cognitive prospect have to be considered aimed at bridging the gaps between software and cognitive sciences regarding knowledge. Finally, Meso et al. [5] advocated that the strong software process, tailored for a particular application context, should fitted to cognitive theory.

Despite these different interests and perspectives, it follows that the integration of knowledge component can improves the efficiency of the software processes as well as their quality. However, the latest specification of Software & Systems Process Engineering Meta-model (SPEM 2.0) [6], the OMG‟s "*de facto*" standard devoted to software process modeling, does not supports this concern. It focuses on a structural view and does not define support for such behavior modeling. That‟s why there is a need to extend this Meta-model to support a knowledge-oriented modeling perspective on the base of activity-oriented one.

A typical problem faced by project managers when starting a software project, either new or maintenance, relates to the question: Do we have necessary knowledge to complete the project? Data is required to support an informed decision: For all interrelated activities, which are the unit of work of a given process, is it possible to measure the knowledge required to carry out each task and to map this data to knowledge provided by Roles (primary and additional) as well as input artifacts ? Hence, there is a need for a dashboard that would helps to develop indices of knowledge discrepancies. So we propose a formalism that is based on: 1) the SPEM standard, which is used for building the syntactic structure, and so providing a standardized static structural view; and 2) an extension based on the relationships between components of that structural view, which is used to formalize the semantic relationships between SPEM elements, and so supporting a conceptual view of Knowledge. This formal approach allows process designers to create, as well as to represent, analyze and validate a Knowledge view of process model.

To demonstrate the potential of this conceptual Knowledge-oriented approach, this paper presents a new perspective for software process modeling supported by a new tool called DSL4SPM (Domain-Specific-Language for Software Process Modeling). This approach may be addressed by adding a new perspective on top of the Activity-oriented one. Our proposal consists to enrich the semantics of a process model by exploiting the relationships between SPEM elements. Thus, we can address multiple views based on these relationships.

The remaining of the paper is organized as follows: Section 2 outlines the background theories. Section 3 highlights the architecture of DSL4SPM tool and the basis of the proposed approach. Section 4 illustrates an example based on post hoc analysis of the process data related to a real project. Section 5 presents the conclusion and future works.

2. Theoretical Background

2.1 SPEM 2.0 Overview

SPEM2.0 is the OMG‟s standard aimed at software process modeling [6]. It is based on UML and dedicated to describe components of software process. The Meta-model separates the content of methodology from its instantiation within a particular process as shown in Figure 1. This improves the reuse of predefined elements, such as Role, Task, Work Products and Guidance in specific processes customized to specific types of projects.

According to SPEM, a disciplined process is a set of activities; each activity is composed of one or more tasks performed by an abstract active entity called Role that is responsible for one or more tangible entities called work products. Activities are organized within phases. Tasks can be more detailed with Steps. Roles describe the responsibilities and a set of skills used to make easy the assignation to reel persons. Work products are a piece of information produced or used by the tasks.

SPEM 2.0 is activity oriented Meta-model, it is founded on basis of seven packages: 1) The Core contains common classes and abstractions used by all other packages; 2) Process Structure defines and represents a breakdown structure of nested Activities. Activity references list of tasks, roles and Work Products, which are defined in Method Content package; 3) Process Behavior allows extension of SPEM Meta-model toward existing externally-defined behavior models; 4) Managed Content describes in natural language documentation of model components (ex. White paper describing agile practices); 5) Method Content describes element such as Roles, Tasks, and Work product Definitions; 6) Process with Methods uses the elements already defined in Method Content package to build a specific lifecycle or breakdown structure of activities for a specific project; 7) Method Plug-in is a repository for configurable libraries, which packages Method content and processes contents to be reused.

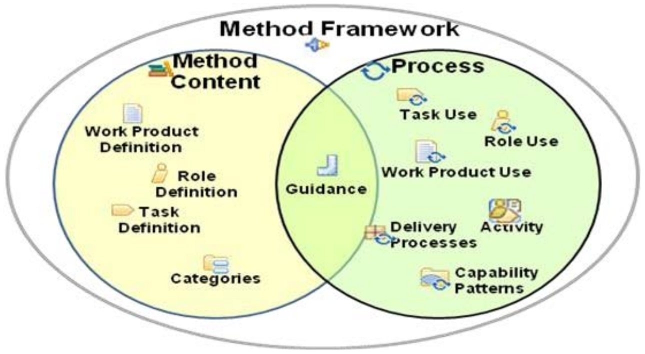


Figure. 1.SPEM Method Framework mapped to Method Content versus Process REF [6].

Figure 1 shows the synthesis of the two most interesting packages for us: The Method Content and Process packages. While there is a great acceptance regarding SPEM 2.0 as Meta-model for software development processes, one important basic concern is the need of tools that support a conceptual approach from which other perspective such knowledge-oriented can be integrated to processes modeling.

2.2 Knowledge Management theories

According to Davenport and Prusak [7], Knowledge Management (KM) can be defined as a process that supports sharing, distributing, creating, capturing and understanding organization‟s knowledge. This definition seems to be general and can be separated on a number of issues: knowledge creation, representation and sharing. This subsection highlights what we think are the representative theories coming from both management and cognitive sciences. The management theories emphasizes on organization supporting knowledge creation and propagation, while cognitive theories emphasizes on knowledge representation and storage.

From management point of view, Nonaka & Takeuchi [8] present their knowledge-creation theory which is represented in a bi-dimensional plan: the first is epistemological dimension that emphasizes on two types of knowledge: Tacit (T) and Explicit (E). Tacit knowledge is personal and context-specific wherein by what a person is not able to express explicitly but which frames his behavior, while the explicit knowledge is an externalized knowledge that can be represented in formal or informal support of communication independently of who “knows”. For the software processes, this means that knowledge may exist in a personal (Role) and in registered format (Artifact). The second is ontological dimension that emphasizes on organizational/managerial structures (i.e. Knowledge level): Individual, group, organizational and inter-organizational. Nonaka & Takeuchi argue that knowledge creation is fostered by a conversion process transforming tacit knowledge to explicit knowledge, and vice-versa, throughout a hierarchical structure (i.e. ontological dimension). They identified four cycles of conversion: Internalization (E-To-T), Externalization (T-to-E); socialization (T-To-T) and combination (E-To-E).

Arguing that the important characteristic of knowledge relates to action, many authors [8-10] recognize that a significant implication of knowledge representation is to achieve a same understanding of data or information within share of certain knowledge base. However, there is a lack for formalism on how the knowledge can be embodied within a process model conceptually and not by a simple text description. Hence, there is a need for a symmetric representation of knowledge, which means that we could use the same descriptive object for roles as well as tasks or artifacts.

Unlike management science, cognitive science looks at how human mind stores things. It defines knowledge as a set of interrelated concepts [11]. Concepts, which are defined as the basic unit of knowledge, are at the center of theories for knowledge representation [12]. Those concepts can be organized in ontology to simplify their manipulation as well as to share meanings and semantics. We believe it would be useful to represent knowledge as it is represented in the human mind (i.e. a set of concepts).

From cognitive point of view, Novak and Canas [13] developed the theory of knowledge representation and/or sharing systems, including individual learning. The authors defined knowledge as a structured set of interrelated concepts (concept mapping). This approach is used to represent as well as to share the knowledge on a specific domain. Novak argues that learning involves the assimilation (i.e. internalization) of new concepts into existing cognitive structures as recommended by the "Constructivist learning theory". Indeed, cognitive psychology stated that people learn by the assimilation of new concepts and propositions into existing cognitive structures, instead memorizing [14].

Recognizing that there are two types of knowledge : Declarative and Procedural, Anderson [15] assumes that the Declarative knowledge refers to things that can be described and shared with others (e.g., answer a question about a programming language), while Procedural knowledge is the knowledge exercised in the performance of a task and focuses mostly on action than information. This second type of knowledge is difficult to describe, nevertheless, important particularly in problem solving (e.g. experience of using a debugger).

Even though there is an abundance of theoretical knowledge models in the literature, there is still a lack of integrated tools for the KM as reported by Bjørnson & Dingsøyr in their systematic review [16]. Recognizing that knowledge is the primary source of an organization's innovative potential, this work intends to integrate knowledge-oriented perspective on basis of activity-oriented perspective aiming at represent and manage knowledge within software development processes. First, SPEM Meta-model is extended with attributes related to knowledge. So a cognitive approach is adopted to represent the knowledge required to carry out a task (considered as the core of action) and symmetrically the knowledge provided by artifacts and roles that are linked to this task. Second, the knowledge required and provided are compared and system generate a dashboard. The next section presents the foundations of the tool used to implement this perspective and describes the steps of the proposed approach.

3. Extends conceptual modeling with the Knowledge Representation

3.1 The system architecture of DSL4SPM tool

DSL4SPM target the compliance point called “SPEM Process with Behavior and Content” [6]. This compliance point is recommended for implementers who want to focus on the modeling aspects of SPEM. This subsection presents the architecture, which has been used for DSL4SPM tool that supports the proposed extension.

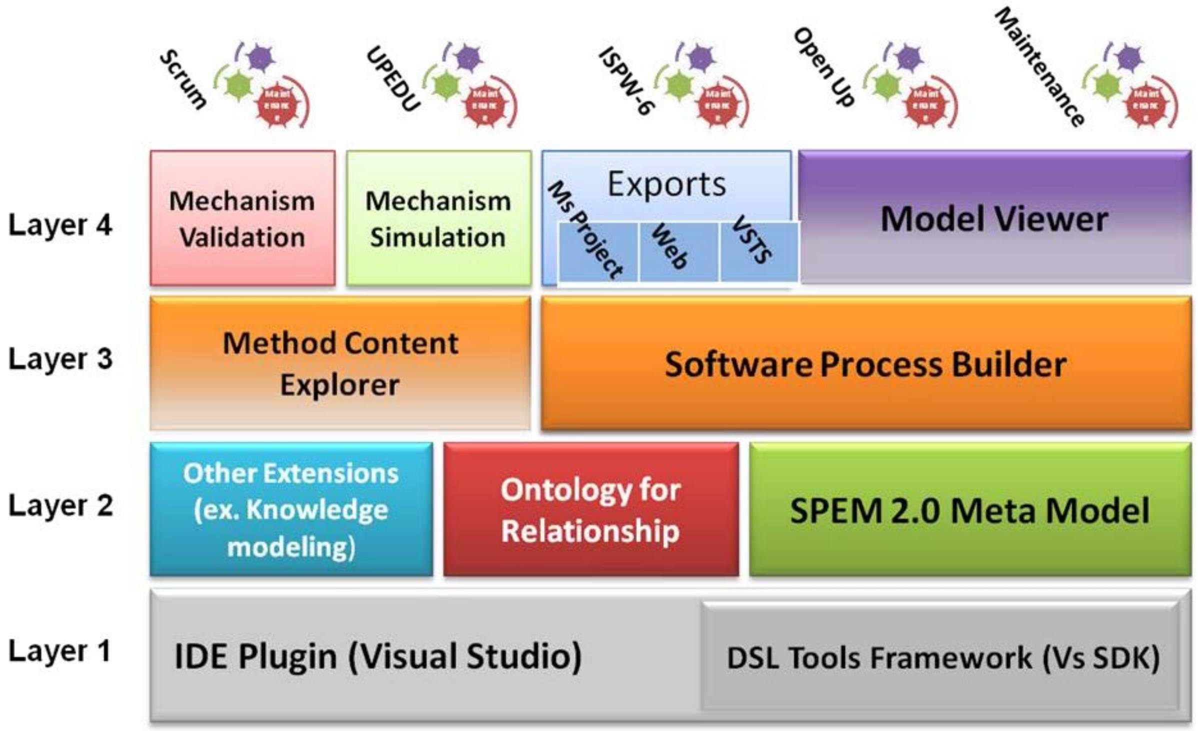


Figure 2.Architecture of the DSL4SPM tool

- Layer 1 – This lowest level represents the technical infrastructure chosen for the DSL4SPM tool implementation and persistency services. The choice of Visual Studio IDE is based on two key points: 1) Using C # language which provides the mechanism of partial class. This mechanism allows separation of one class implementation in several physical files (graphics, specific code, validation, etc.); 2) Provides "DSL Tools" framework that supports the creation of a graphical Domain-Specific-Language, which helps improving the semantics of SPEM with a specific language based on the concept of attributed relationships. It means that the focus is not just about the structure of a process but also on information contained in the attributes of the relationship between SPEM elements and allows for multi-view representation. The next section presents the ontological formalism of these relationships.

- Layer 2 represents the Meta-model SPEM 2.0. Since this Meta-model does not support the modeling of behavioral aspects, it has been extended with a Role-Activity-Artifact ontological formalism. This component allows the validation of the model consistency (e.g. Verify the kind of links, which can be performed, between two given elements). Others extensions are proposed, their detail is out of scope of this paper.

- Layer 3 represents the tool interface that supports the modeling (e.g. Tool Box, Scene, Breakdown structures and a specific component supporting the communication with a Method content, a repository that describes reusable elements).

- Layer 4 represents the usability of a process model, which address, on top of Activity-Oriented structure, others Breakdown structures such as knowledge which focuses on tacit and explicit flows throughout the process. It can also be exported to a Website, Visual Studio Team System (VSTS) and project management tool such as Ms Project. Due to the unpredictable nature of task‟s duration, the process can be simulated from the estimated duration of tasks and their sequence embodied in the link Task-to-Task. The component of validation can be used to check the consistency of a process model according to predefined rules (e.g. a primary role for each task, a work product should be related to an activity, etc.).

3.2 Basis of the approach

Based on the DSL4SPM tool architecture as described in preceding subsection, the knowledge-oriented perspective has been developed on top of activity-oriented perspective.

Figure 3 depicts the approach of dashboard building aiming at visualize the mismatch between the knowledge required to carry out situational task and other SPEM elements (e.g. Role, artifact, guidance, etc.) that provide this knowledge. The build of dashboard follows three steps:

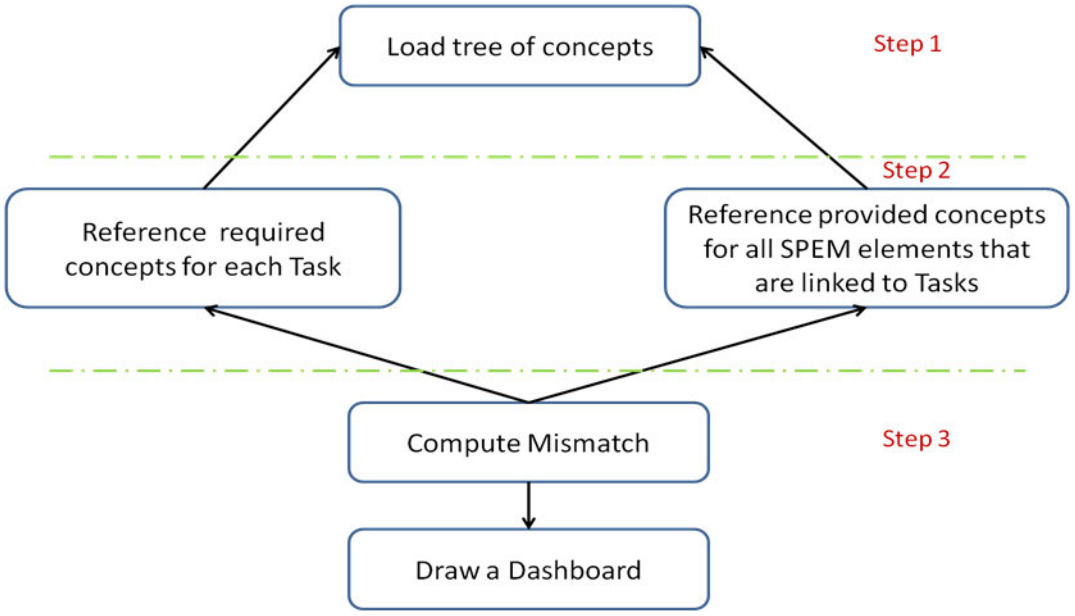

Load tree of concepts
Step 1
Step 2
Reference required concepts for each Task
Reference provided concepts for all SPEM elements that are linked to Tasks
Compute Mismatch
Step 3
Draw a Dashboard

Figure 3. The dashboard building approach.

STEP 1: Parameterization

- This step specifies the ontology that formalizes the structure of concepts. The process engineer loads a default or adapted tree of concepts (i.e. ontology) from a structured XML file. The project team gathers and organizes concepts of knowledge relevant to the context of project. As defined by [17], an ontology is a formal description of knowledge aiming at provides a common understanding of topics to be shared between users and systems. Following this definition and based on work of Anquetil et al. [18], ontology of concept has been used for materialize knowledge-sharing. As seen in figure 4 (f), a tree of concept is proposed by default. This tree is recorded in XML format, which means more flexibility for adjustment to a specific context of process and/or project.

STEP 2: Modeling

- For each task, set references to a subset of concepts required for achieving the task. For each concept, the designer specifies in which way the concept is required declarative and/or procedural.

- For all incoming links to each task (e.g. from Work products, roles, guidance), set references to a subset of concepts which are provided by this element.

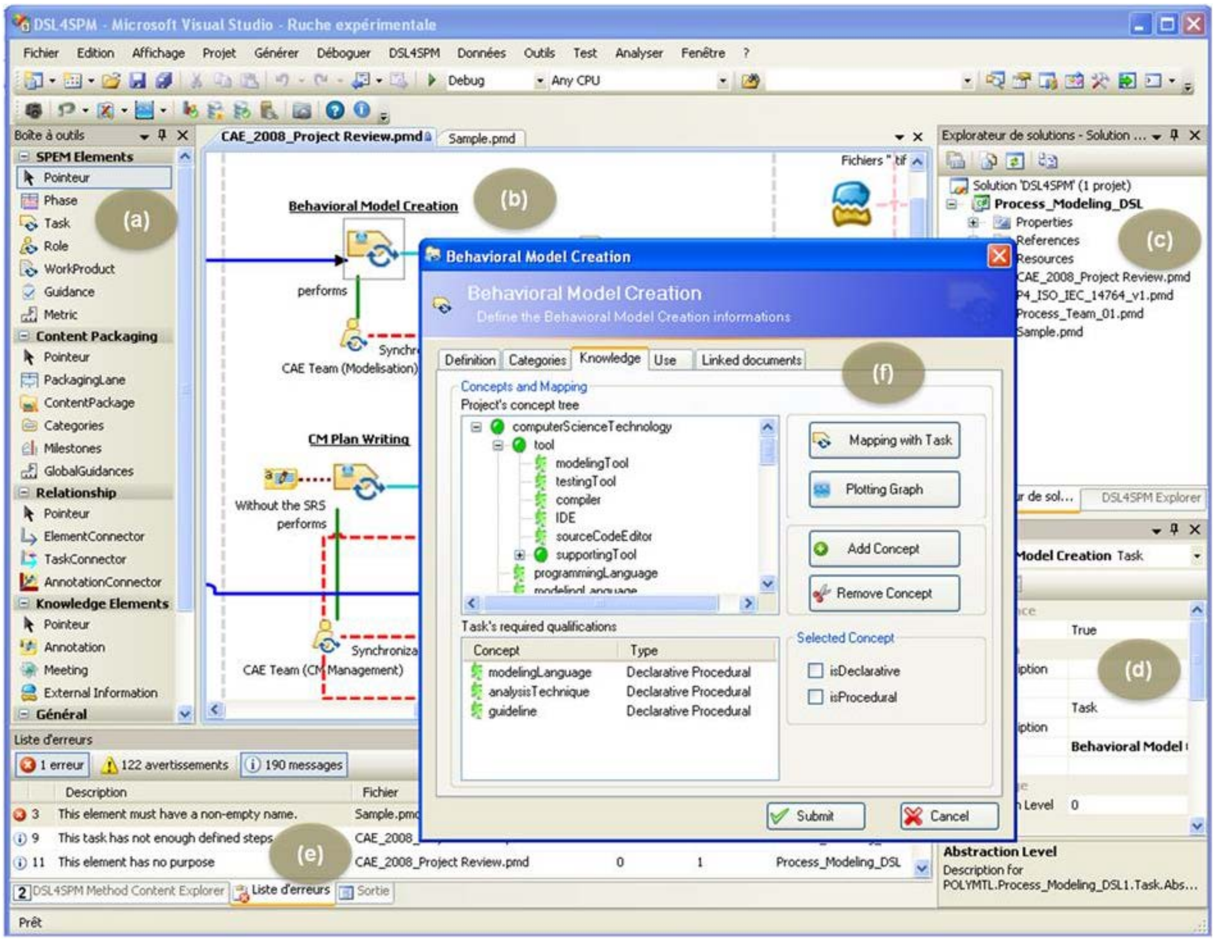
STEP 3: Compute mismatch and draw a dashboard

- For each task, the system searches all incoming links, retrieves the concepts provided and compiles the results. Thus, a concept required by the task is considered as fully mapped if it can be provided, with the proper type, by at least one of the elements that are linked (incoming) to this task. A concept required by the task is considered inadequately mapped if it is partially mapped (ex. provided as declarative, while it is required such as procedural). Finally, a concept is considered non-mapped if it isn‟t provided by any SPEM elements linked to this task.

■ Dashboard displays the result of „mapping concepts‟ (as seen in figure 5 below), which depicts the mismatch of knowledge. In doing so, the user can visually note how much the needs, in terms of knowledge, diverge from an adequate situation.

4. Tool illustration, contributions and limits

In order to validate the benefits of this approach, an example based on post hoc analysis of the process data related to a real project has been modeled with DSL4SPM tool. The partial result is shown in Figure 4.



Concept	Type
modelingLanguage	Declarative Procedural
analysisTechnique	Declarative Procedural
guideline	Declarative Procedural

	Description	Fichier			
3	This element must have a non-empty name.	Sample.pmd			
9	This task has not enough defined steps	CAE_2008_			
11	This element has no purpose	CAE_2008_Project Review.pmd	0	1	Process_Modeling_DSL

Figure 4 the DSL4SPM modeling environment. Knowledge perspective is highlighted.

Figure 4 presents the DSL4SPM modeling environment and highlights six main areas:

■ **(a)** DSL4SPM Tool box, which contains the instantiable elements classified in groups such as SPEM elements, Content packaging, Relationships and Knowledge elements. Each group contains process components that the process engineer can drag-and-drop to the scene of modeling.

■ **(b)** DSL4SPM modeling scene which represent a model of process.

■ **(c)** Visual Studio navigator view that represents a project of process modeling composed of one or more process model.

■ **(d)** Visual Studio properties window that represents the attributes of each element within the scene of modeling.

■ **(e)** Visual Studio problems view, which is use by DSL4SPM to raise exceptions according to a formal verification against established rules of coherence.

■ **(f)** DSL4SPM specific form for each element within modeling area. The form is organized according to predefined views. Knowledge is an example of view, which represents a knowledge-oriented perspective rather than activity-oriented one.

Figure 5 shows a dashboard generated by the system following the mapping of knowledge concepts between each Task and other SPEM elements linked to it. This dashboard represents the tasks and their knowledge status with "*CompletelyMapped*", "*InadequacyMapped*" and "*Unmapped*" indicators to remind the management team of which tasks are not risky and which one remain to be analyzed.

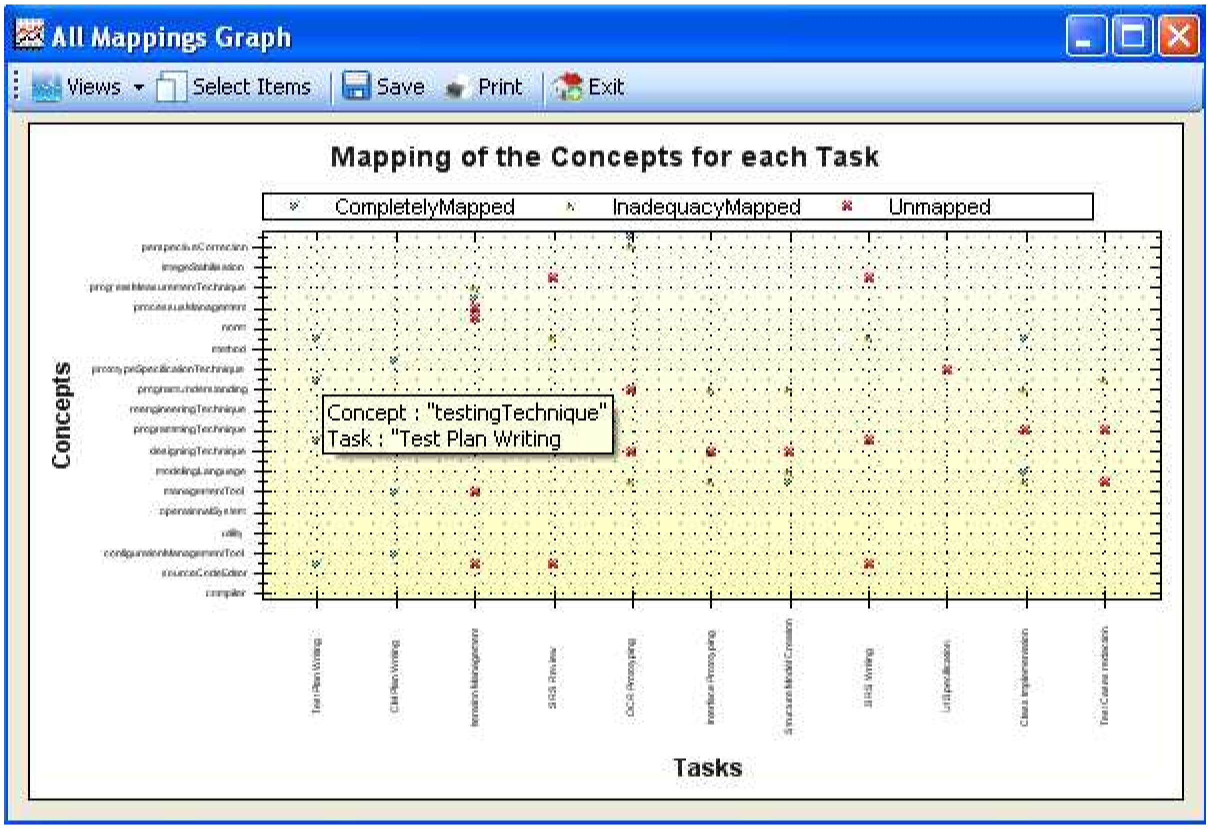


Figure 5 System dashboard generated from concepts mapping.

Table 1 summarizes quantitatively the concept mapping for all tasks within process, number of SPEM elements linked to each task, number of concepts required to carry out the task and number of concepts provided by the elements linked to the task.

Table 1. Synthesis of Knowledge concepts mapping

Task	Links in	Concepts required	Provided concepts
Iteration Management	4	6	4
SRS Review	1	3	2
SRS Writing	3	4	3
UI Specification	0	1	0
Behavioral Model Creation	2	3	4
Test Plan Writing	1	4	4
CM Plan Writing	3	3	6
OCR Prototyping	6	6	8
Interface Prototyping	5	4	6
Structure Model Creation	10	4	9
Class Implementation	6	5	10
Test Cases Redaction	4	3	3
UI Implementation	5	4	9

The approach proposed, specifically the dashboard, might assist the manager team to make an informed decision:

- The project manager could analyzes the competence gap which indicates any discrepancies between the knowledge concepts required for each activity and the aggregated concepts provided by all the SPEM element around this activity (e.g. Work products, guidance, roles). In doing so, the system highlights the problematic tasks that need more knowledge concepts and be able to

- Search for SPEM elements that can provides the missing concepts.

- Make a contingency plan which could be to identify a resource (interne or extern) that could bring some expertise.

- Add additional roles (team up) to support primary role

- Change strategy for risk migration based on prototyping or on short iterations.

- Reorganize the process activities or project schedule to avoid the risk.

- Allocate the ownership of part of project to a third party who is best able to carries out the opportunity.

The system provides dashboard as rich pictures that supports managers for decisions like external recruiting, forming teams, support for organizational activities such as availability planning, support for searching for and find a role with specific kind of knowledge.

4.1 Limits of the proposed solution

At the beginning of the project realization, it might be difficult to assess the needs in terms of knowledge due to limited vision of the solution, which is the same difficulty seen for planning and estimation.

• Like the expert system, it could be necessary to have an intervention of domain expert early in the project to adapt the ontology of knowledge (i.e. tree of concepts) to specific local context of the project.

• The relevance of each concept of knowledge to carry out task on hand has been cut back to minimize the complexity of use.

• Empirical tests are needed to refine the proposed ontology as well as to validate the usefulness of the approach within industrial context.

5. Conclusion and Future Works

We have proposed a method to put forward a process-oriented knowledge and a technique based on concept mapping to represent knowledge in the process model. This knowledge-oriented approach extend SPEM 2.0 and is supported by DSL4SPM, a tool that supports software process modeling, based on SPEM for the syntactic definition and semantic extension that exploit the relationships between SPEM elements.

This method highlights the importance of knowledge modeling, while the development of a dashboard for visualizing knowledge mapping is the major achievement. The introduction of knowledge concepts in software processes mitigates the risks related to knowledge mismatch and supports an informed decision early in the lifecycle of project. This knowledge mapping view assists the project managers to identify the tasks that need more knowledge concepts. After analyzing the mismatch of concepts, the project managers can take rational and efficient decisions such as recruitment of new competences or adding additional roles (team up) to support primary role performing the task on hand. Hence, the knowledge-oriented view complements activity-oriented view and thereby fosters a better understanding of complex processes by emphasizing appropriate abstractions.

The efficiency of modeling and visualization approach has been illustrated throughout post hoc analysis of the process data related to project developed by a team of undergraduate students. The results showed that the approach is capable to represent a useful dashboard that is helpful to understand the needs of knowledge for each Task.

Future work will include a more detailed analysis of whole process related to knowledge flows among software process model within an industrial project. The targeted goal will be assessment of the propagation of knowledge throughout all the phases of software process aiming at supporting the project managers to identify the problematic arcs.

6. ACKNOWLEDGMENTS

The first author would like to thank Olivier Gendrou for helping with the experiment data, and analyses. Part of the work presented was financially supported by the “Fonds de Recherche sur la Nature et les Technologies” council of Québec (FQRNT) under Grant 127037.

7. REFERENCES

[1] S. Dakhli and M. Ben Chouikha, "The knowledge-gap reduction in software engineering," Piscataway, NJ, USA, 2009, pp. 287-294.

[2] I. Rus and M. Lindvall, "Knowledge management in software engineering," Software, IEEE, vol. 19, pp. 26-38, 2002.

[3] X. Junchao, et al., "Applying little-JIL to describe process-agent knowledge and support project planning in soft PM," Software Process: Improvement and Practice, vol. 12, pp. 437-48, 2007.

[4] P. N. Robillard, "The role of knowledge in software development," Communications of the ACM, vol. 42, pp. 87-93, 1999.

[5] P. Meso, et al., "The knowledge management efficacy of matching information systems development methodologies with application characteristics-an experimental study," Journal of Systems and Software, vol. 79, pp. 15-28, 2006.

[6] SPEM. Software & Systems Process Engineering Meta-Model Specification. Version 2.0. Final Adopted Specification.

[7] T. H. Davenport and L. L. Prusak, Working Knowledge: How Organizations Manage What They Know. Boston, USA: Harvard Business School Press, 1998.

[8] I. Nonaka and H. Takeuchi, The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation: Oxford University Press, USA, 1995.

[9] M. Alavi and D. E. Leidner, "Knowledge management and knowledge management systems: conceptual foundations and research issues," MIS Quarterly, vol. 25, pp. 107-36, 2001.

[10] M. L. Markus, et al., "A design theory for systems that support emergent knowledge processes," MIS Quarterly, vol. 26, pp. 179-212, 2002.

[11] W. Yingxu, "On concept algebra and knowledge representation," Los Alamitos, CA, USA, 2006, pp. 320-231.

[12] J. R. Anderson, The Architecture of Cognition: Lawrence Erlbaum; Reprint edition (November 1, 1995), 1983.

[13] J. D. Novak and A. J. Cañas, "The Theory Underlying Concept Maps and How to Construct Them," Technical Report IHMC CmapTools 2006-01 Rev 01-2008, Florida Institute for Human and Machine Cognition, available at: http://cmap.ihmc.us/Publications/ResearchPapers/TheoryUnderlyingConceptMaps.pdf, 2008.

[14] D. P. Asubel, The psychology of meaningful verbal learning. New York: Grune and Stratton, 1963.

[15] J. R. Anderson, et al., "ACT-R: a theory of higher level cognition and its relation to visual attention," vol. 12, pp. 439-462, 1997.

[16] F. O. Bjørnson and T. Dingsøyr, "Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used," Information and Software Technology, vol. 50, pp. 1055-1068, 2008.

[17] N. R. Noy, "Semantic integration: a survey of ontology-based approaches," SIGMOD Record, vol. 33, pp. 65-70, 2004.

[18] N. Anquetil, et al., "Software maintenance seen as a knowledge management issue," Information and Software Technology, vol. 49, pp. 515-29, 2007.