Spemmet - A Tool for Modeling Software Processes with SPEM

Tuomas Mäkilä *
tuomas.makila@it.utu.fi

Antero Järvi*
antero.jarvi@it.utu.fi

Abstract: The software development process has many unique attributes and therefore the modeling language should be designed for this particular task. The emerging Software Process Engineering Metamodel (SPEM) modeling standard from the Object Management Group (OMG) offers one solution of modeling software processes. In this paper the Software Process Engineering Metamodel is presented in brief and challenges on implementing a SPEM based modeling tool are discussed. Especially the Spemmet process modeling tool implemented by the authors is presented and analyzed.

Keywords: software process modeling, SPEM

1 Introduction

Business process modeling is a tool in information system development for understanding the dynamic behavior of the system context. Similarly, the dynamic complexity of information system development is managed by modeling the underlying development processes. However, these processes have different characteristics than business processes and thus can not be modeled well with the present business process modeling languages e.g. traditional flow chart notation, *Business Process Modeling Notation (BPMN)* [1], *Integrated Definition Methods (IDEF)* [2], or *Event-Process Chains (EPC)* [3].

The development of information systems is a highly complicated task, involving coordination of actions of many people working under uncertainty, managing constant change, and at the same time using resources efficiently to meet tight deadlines. Software processes need to balance predictability and efficiency with flexibility of operation and creativity. The stable elements in software development are the work products; how created value is captured to work products during the project is less volatile than the activities and their order of execution. Therefore well-known business process modeling languages based on activity oriented modeling fail to express the required flexibility in modeling processes characterized by uncertainty and creativity.

Process modeling languages that are built around the work products, so called entity-based modeling languages, are more suitable for modeling software development processes. One such language is *the Software Process Engineering Metamodel (SPEM)* from *the Object Management Group (OMG)* [4]. The SPEM is based on *the OMG Meta Object Facility (MOF)* [5]. The purpose of the SPEM is to become the standard software process modeling language. The current version of the SPEM is 1.1 which was released in January 2005. In a few years there will be a major revision of the standard [6].

The research of software process specific modeling languages has been mild during recent few years. Some work on the topic has been done. García et al. have developed metrics for

* University of Turku, Department of Information Technology, FI-20014 Turku, Finland

evaluating maintainability of the software process models [7][8]. They have used the SPEM as a primary modeling language. Franch and Ribó have investigated ways to enhance reuse with appropriate modeling conventions [9]. They have developed own UML-based modeling language PROMENADE to support the reuse.

Besides the academic research there are some commercial tools, that support software process modeling techniques. Key practitioners in this field are IBM and Osellus. IBM provides a software process modeling tool that supports their Rational Unified Process framework [10]. Osellus have developed the IRIS product family which is meant for process authoring and enactment [11]. The IRIS product family has been built on the SPEM modeling language.

One interesting project related to the topic is the freshly started Eclipse Process Framework Project (EPF) [12]. The goal of the project is to provide process authoring tool framework and process content for different development needs. The work is based on the open Eclipse platform.

This report is based on the ongoing ReProCo research project, that investigates how software development should be supported by process modeling. The central issues are the type of modeled process content, its organization into process components, and the mechanisms of process content reuse. The aim is to provide flexible and customizable process models that can meet the high variety of the process needs, which are typically found in software companies. In addition these process models should be light to define and maintain. The concepts of the process content organization into reusable process libraries are presented in another paper [13].

This paper will focus on describing technological elements of software process modeling: the modeling language and a modeling tool based on the language. The context of software process modeling should still be kept in mind. The process models should be organized into reusable process libraries, which can be selected, tailored, and enacted by software companies based on the organizational and the project needs.

2 Software Process Engineering Metamodel (SPEM)

2.1 SPEM standard

SPEM standard is closely related to *the UML* standard, since they both are based on the MOF specification. The SPEM 1.1 standard is actually built onto the SPEM Foundations package which is a subset of the UML 1.4 [14]. Basic element, relationship, and package structures are defined in this Foundations package. All other SPEM elements are defined in relation to the elements of the Foundations package through inheritance. Because there is a close connection between the SPEM and the UML, it is natural that the SPEM is also defined as a UML profile in the SPEM documentation [4, Chapter 11].

Although the SPEM notation resembles the UML, it has otherwise quite different structure. At the conceptual level, the main idea of SPEM based process is the interplay of three basic elements: *Process Roles* that are responsible for and execute *Activities* that consume and produce *Work Products*. Roles, Work Products, and Activities are all process definition elements. This is illustrated in Figure 1.

SPEM defines elements *Life Cycle*, *Phase* and *Iteration* that are used to model the dynamic structure of the process. A Life Cycle defines the order of Phases, which in turn can contain Iterations. A Process must have exactly one Life Cycle.

SPEM also defines elements that are meant for organizing other process elements from the viewpoint of process authoring, assembly and reuse. The purpose of *Packages* is to divide

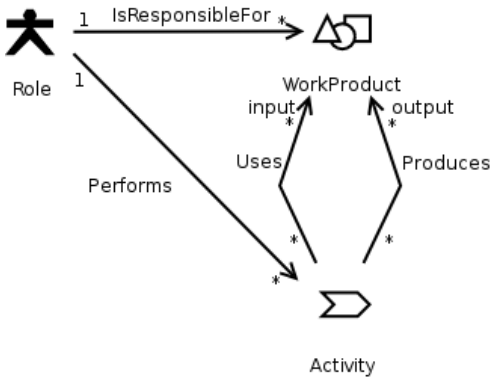

Fig. 1: The core concept of SPEM process modeling is a triangle formed by Role, Work Product and Activity.

process descriptions into self-containing parts. These parts can then be placed under configuration and version management and used for assembling and tailoring software development processes. *Process Components* are specializations of Packages. A Process Component is an internally consistent and self-contained chunk of process descriptions that may be reused with other Process Components to assemble complete processes. Process Components can import a non-arbitrary set of process definition elements. The *Discipline* is a specialization of the Process Component and is used to represent activities within a common process area, such as design, implementation or testing (corresponding to e.g. Core Work-flows in the Unified Process [15]). *Process* itself is also a specialization of the Process Component, that is intended to stand alone as a complete end-to-end process.

The assembly of processes is done by composition of Process Components. This requires unification of the Process Components. Corresponding output and input Work Products must be unified, as well as Process Roles and possibly other elements that are used in more than one Process Component. The details of unification are not defined in SPEM.

2.2 The scope of process modeling

Process modeling can be used in many different scales; at its simplest, an existing software process in a company would be modeled for clearer presentation and easier delivery. On the other extreme, a large company might manage a library of software process content and new end-to-end processes would be created by combining and customizing process components, either internally developed or third party components. It is evident, that the more complex applications of process modeling and reuse require more sophisticated mechanisms for organizing process content. Unfortunately SPEM 1.1 cannot provide these mechanisms. However, the core ideas of SPEM modeling are unlikely to change. For this reason we are currently targeting for simple applications of process modeling; having only internally developed process content, a clear separation of static and dynamic process content, and simple mechanisms of process assembly [13]. As the SPEM standard evolves, these basic modeling concepts should remain valid.

3 Spemmet Process Modeling Tool

In this section we present a SPEM based modeling tool called *Spemmet*. The development of the tool begun at the early stage of the ReProCo project when we had to form a prototype process component model using the SPEM modeling language.

Before the decision of the own SPEM modeling tool was made, several readily available tools were evaluated. The evaluated tools can be divided into four categories: Basic drawing tools, UML tools, XML tools, and SPEM specific editors. Drawing tools, UML tools, and SPEM tools concentrated on diagram drawing, whereas we needed more efficient means to bind textual data (e.g. tailoring guidelines) to process elements. These tools also had limited modeling support for our needs. XML tools were evaluated in order to exploit the SPEM XMI DTD (XML Metadata Interchange Document Type Definition) provided by OMG [16] for manual data input. It soon become apparent that the DTD was meant only for exchanging information between modeling programs and could not be used as a base for forms for manual data entry [1].

When the Spemmet development started we made a couple of key design decisions. Firstly, we wanted to make sure that the tool would not restrict our modeling options. Of course, the tool should offer all necessary elements for modeling software processes and support the modeling efforts in order to make the modeling as easy as possible. It is equally important that the tool is flexible and allows the user to use different modeling techniques, because modeling needs change depending on the modeled process. Modeling tool must also allow user to experiment with different modeling techniques since there are no established software process modeling guidelines available.

Second design decision was to omit graphical representation of the process model and allow user to input and output data into the model only in textual form. It was apparent that the textual list of the attributes of a process element was easy to comprehend. Actually it would have been difficult to clearly express all attributes of one element in a graphical diagram. After the process element data is entered into machine-readable form, it is relatively easy to generate various graphical and textual views from this data in future version of the tool. This is an important design aspect, because in practice different stakeholders have different purposes of use for the same process model, and logical way of taking these varying and currently unanticipated requirements into consideration is to be able to offer multiple views to the model.

The implementation of the Spemmet tool was quite straightforward. It was developed on a web platform and used a shared data storage. The idea was that the tool would be available on any workstation without installation and multiple users could use it simultaneous. The overview of the architecture of the tool can been seen in Figure 2.

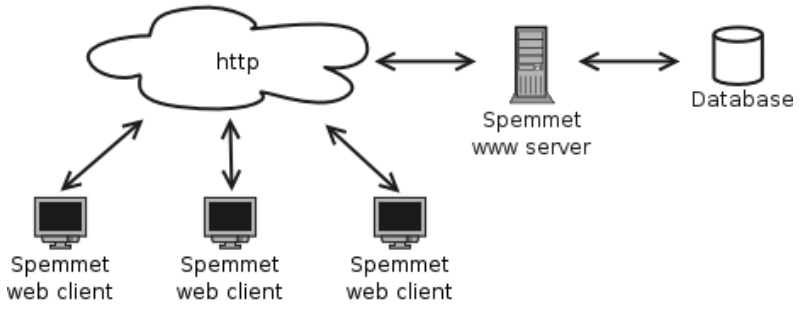

Fig. 2: Overview of the Spemmet architecture. Spemmet web clients communicate with the Spemmet server through the http protocol. The model data is stored in the database.

The most difficult task during the implementation was to transform the conceptual model of

[1] This was an anticipated outcome, because the original purpose of the XMI definitions is to enable metadata interchange between modeling tools.

the SPEM to the actual data structure definitions used by the tool. This was a consequence of the complexity of the SPEM standard specification. Some parts of the standard were ambiguous or simply too complex to be feasible in this kind of tool prototype. Good example of the latter is the state machine described in the SPEM Foundations package, which might be good for the UML 1.4 but is too complex for the SPEM.

Luckily the UML Profile definition of the SPEM was easier to understand and helped resolving some of the metamodel ambiguities. Some shortcuts were made with the most complex parts of the SPEM metamodel. However, the overall data structure of the Spemmet tool was kept as compatible with the SPEM standard as possible.

The most important classes of the Spemmet tool data model are presented in Figure 3. Only structures used for modeling static process elements were implemented in this first version of the Spemmet tool. Therefore the basic conceptual triangle of the SPEM presented in Figure 1 formed the heart of the class structure of the Spemmet tool. Guidance and Package classes were adopted to support the process reuse: The Guidance class enabled documentation and tailoring guidelines integration into the process models and the Package class made possible to logically group process model elements. Figure 3 also shows that all of the SPEM classes are inherited from the Model Element class.

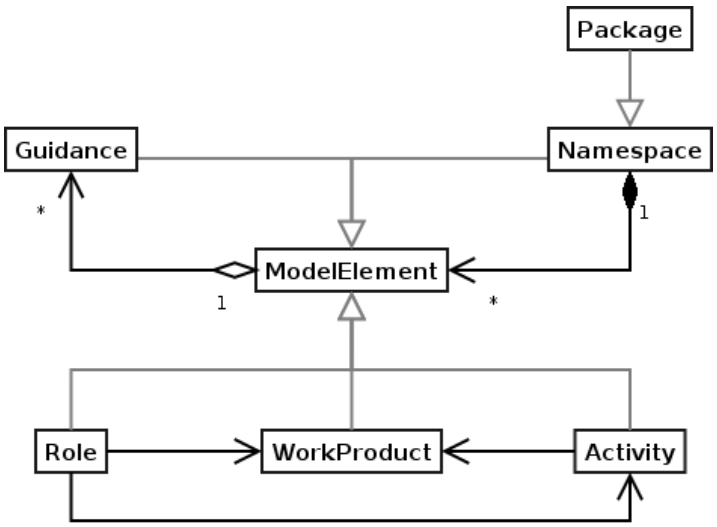

Fig. 3: Core classes from the SPEM implemented in the Spemmet tool.

The first version of the Spemmet tool was successfully implemented and included all the features that were needed in our modeling project. The basic architecture works fine and it is possible to use modeling tool with a web browser through any workstation in our intranet. Multiple users are able to model with the tool simultaneously, although there are still some scalability issues and no user management in this first version.

The most important feature for flexible and practical process modeling styles is the versatile linking between the process elements. All basic elements can be linked together using multiple relationship types. Association, dependency, and inheritance, which are familiar from the UML standard, are all supported. The user can to browse elements by following these relationship links. Together with the basic package structure the user can navigate through the process model easily and in appropriate manner.

The user interface of the Spemmet tool is naturally quite simple, because in the SPEM all process elements are inherited from the same parent class (i.e. Model Element class). Therefore the user can browse and edit all process elements in consistent one-screen view. This simple design allowed us also to implement an html exporter, which is a great help when we have to share snapshots of the model with other stakeholders. A screen shot of the Spemmet user interface can be seen in Figure 4.

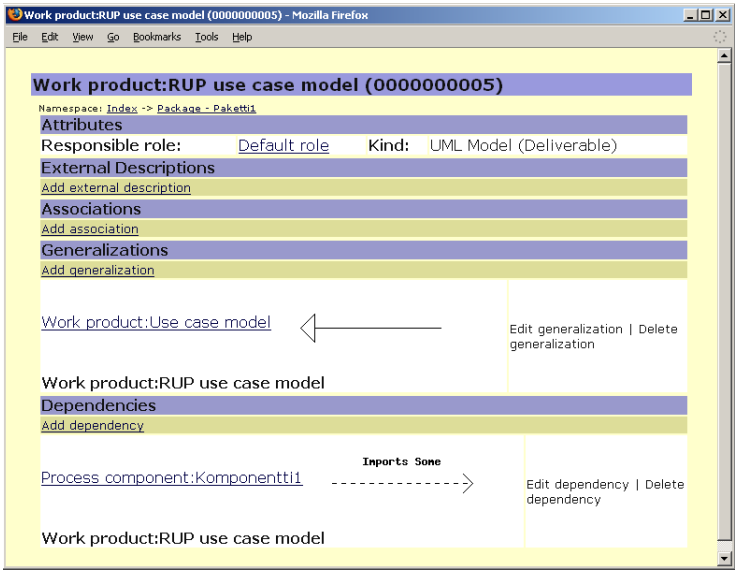

Fig. 4: Screen shot of an element view of the Spemmet tool.

4 Further Work

The Spemmet tool is our first attempt to get practical understanding of the SPEM based process modeling. At the time being the Spemmet tool includes only static process elements of the SPEM standard. The tool must be further developed so that the process dynamics mentioned at the introduction section can be modeled. Especially elements in Process Life Cycle package (i.e. Phase, Life Cycle, Iteration) are important, because these elements are used to express temporal relationships between activities. Modeling of the dynamic structures can however be based on elements that are already included in the tool. For example an activity can be automatically connected to the compatible activities using the input and output work products and their states.

Our work continues on two mutually supporting tracks: first we are carrying out an empirical investigation on the needs and applications of process modeling in software companies. Together with theoretical research on software process modeling, the empirical study results guide the second track, the development of Spemmet tool. The goal is to be able to meet the practical modeling needs in a wide range of companies, and support them with process modeling tools.

The standardization work of SPEM is ongoing; the version 2.0 [6] will be released with in a few years. This of course introduces some uncertainty, but on the other hand, it gives us time to understand the potential and sound ways of applying process modeling in companies.

5 Conclusions

Our experiment with the Spemmet tool shows that it is possible to develop a working process model tool in a relatively short time using the SPEM process modeling standard. The SPEM standard is certainly hard to follow in places and partially too complex. Therefore some shortcuts have to be made at least for this kind of lightweight modeling tool.

The tool was successfully used to model parts of a process framework during the ReProCo project. The modeling was quicker than with tools that were not designed for process modeling, because the tool supported directly SPEM metamodel concepts. The communication about the model itself was also easier with the other stakeholders of the project, because all necessary information of a process element was collected into one place.

The clear benefit from the SPEM specific modeling tool was that the tool makes modeling easier and enables quick modifications. Models made with the software process specific tools seem to be more informative than models ”hacked” with other modeling tools.

The greatest challenge with the SPEM modeling standard is to promote its use both in the academic world and in the software industry. Also, the current SPEM 1.1 version is immature, and this obstacle will probably be removed by the forthcoming SPEM 2.0. The only way we see to increase the adoption of SPEM is to develop working SPEM based modeling tools that will help learning modeling techniques and experimenting with the SPEM.

This article is based on work done during the ReProCo research project (Sub-project of the E!3320 project) in co-operation with Genestia Group Inc. - Neoxen Systems and Devera Software Development Center.

Bibliography

1. *Business Process Modeling Notation Specification - Final Adopted Specification.* Object Management Group, 2006. dtc/06-02-01.
2. *Integrated Definition Methods Home Page.* `http://www.idef.com/`, Knowledge Based Systems Inc. Accessed on March 16 2006.
3. Becker et al.: *Process Management - A Guide for the Design of Business Processes.* Springer, 2003.
4. *Software Process Engineering Metamodel Specification - Version 1.1.* Object Management Group, 2005. formal/05-01-06.
5. *Meta Object Facility (MOF) Specification - Version 1.3.* Object Management Group, 2000. formal/00-04-03.
6. *Software Process Engineering Metamodel (SPEM) 2.0 - Request For Proposal.* Object Management Group, 2004. ad/2004-11-04.
7. García et al.: *Integrated Measurement for the Evaluation and Improvement of Software Processes.* Lecture Notes in Computer Science, Volume 2786, Springer, 2003. pp 94 – 111.
8. García et al.: *Definition and Empirical Validation of Metrics for Software Process Models.* Lecture Notes in Computer Science, Volume 3009, Springer, 2004. pp 146 – 158.
9. Franch and Ribó: *A UML-Based Approach to Enhance Reuse within Process Technology.* Lecture Notes in Computer Science, Volume 2786, Springer, 2003. pp 74 – 93.
10. *Rational Unified Process Home Page.* `http://www.ibm.com/software/awdtools/rup/`, IBM. Accessed on March 16 2006.
11. *Osellus Home Page.* `http://www.osellus.com/`. Accessed on March 16 2006.
12. *Eclipse Process Framework Project Home Page.* `http://www.eclipse.org/epf/`, The Eclipse Foundation. Accessed on March 16 2006.

13. Antero Järvi and Tuomas Mäkilä: *Observations on Modeling Software Processes with SPEM Process Components.* Proceedings of The 9th Symposium on Programming Languages and Software Tools, Tartu, Estonia, 2005.
14. *OMG Unified Modeling Language Specification - Version 1.4.* Object Management Group, 2001. formal/01-09-67.
15. Jacobson et al.: *The Unified Software Development Process.* Addison-Wesley Professional, 1999.
16. *XMI DTD for SPEM 1.0.* Object Management Group, 2003. formal/02-11-14.