An Analysis of Process Support in Knowledge Management Tools for Software Engineering

Torgeir Dingsøyr
Dept. of Computer and Information Science,
Norwegian University of Science and Technology
dingsoyr@idi.ntnu.no

Abstract. Seven tools for knowledge management of experiences from the software engineering domain are analysed to examine the extent to which techniques from the process modeling community are used. We investigate what process model technology is used in the tools, and how these tools represent context. We also examine if the context information used to identify information that is suitable for reuse. Some process elements like information on projects are widely used, as well as support for elicitation. Enactment is supported in one of the seven tools. Context is mostly represented as text, and not with languages that are similar to process modeling languages. Context information is used to a great extent to identify if information is suitable for reuse.

1 Introduction

This paper investigates how process models can be used as a way to add context to experience that is to be reused in software engineering projects. We survey the literature on knowledge management systems that has been developed to support software process improvement and examine to what extent process models are used for harvesting experiences, representing and maintaining them.

When we want to investigate the use of process models for knowledge management in the software engineering domain, it would be too much to look at all relevant factors, like organization of the company, representation of the experiences to be reused, the usability of the tools for knowledge management as well as the project domain.

We limit ourselves to issues that are related to representation in tools that are used, but it is important to have in mind that other factors can be equally important, certainly for an end-user.

Now we organise this paper as follows: First we introduce process modeling and knowledge management, then we construct a framework for analysing what parts and in what way knowledge management systems that are reported in the literature make use of techniques from the process modeling field. We place special emphasis
on systems which are based on the technique Case-based reasoning, as this is considered to be particularly suited to the task \[1\]. We apply the framework for analysis on a selected set of papers from the literature and present it as results, and further discuss the consequences of the results in the discussion.

1.1 Process modeling – an overview

Process modeling is a field within Software Engineering which is focusing on making the process of developing software explicit. Software process models can be used to increase the awareness of how software is developed in an organisation, to change the way software is developed – do software process improvement (this is often referred to as Business Process Reengineering in other fields). Process models can also be used as background to build tools that support software development. This last category of use is called enactment.

When developing process models, several aspects are important:

- **Elicitation** – how do we capture processes that are in use in a company and represent them in a process modeling language? See \[2\].
- **Representation** – what are the needs for the language to represent the process models in? Basic elements are activities, products and resources. See \[3\].
- **Enactment** – how can the process models be used to support software development on the fly? See \[4\].

1.2 A framework for knowledge management

Reusing and handling experiences in companies are often referred to as knowledge management. One way to organise this is by giving the responsibility for capturing and reusing experiences to a separate part of the development organisation.

The experience factory \[5\] is an infrastructure for reusing life cycle experience, processes and products for software development. Experiences are packaged and stored in an experience base.

Experiences should be recorded, generalized, tailored and formalized by an organization that is separated from the project organization. They are also responsible for maintenance. Examples of packages are:

- **Product Packages** - information about the life cycle of a product, information on how to reuse it and lessons learned from reuse.
- **Process Packages** - information on how to execute a life cycle process, and how to reuse.

In order to do the actual reuse, a possible technology is Case-Based Reasoning (CBR). For a further discussion of work on knowledge management in software engineering, see \[6\].
1.3 A framework for analyzing knowledge management systems

With the previous sections in mind, we now go on to define a framework that will be used to analyse some knowledge management systems that are reported in the literature. Our main research question is:

- Are process models suitable as a way to add context to experience that is captured for later use?

To answer this we look at existing tools that are reported in literature, and ask:

- What process model technology is used in the tools that exist now?
- How do these tools represent context?
- Is the context information used to identify information that is suitable for reuse?

From the previous section we can list a set of requirements for process model technology that we will examine if is present in the tools. When we look for these requirements in the tools, we say that they are present if the main function of the requirement is met, and do not take terminology differences into consideration. We group the requirement in three major categories:

* Elicitation – does the tool support elicitation of processes? Is there any support in the tool for capturing and refining processes from software development? We could also say that the tool is dynamic if it supports elicitation.

* Representation – what information related to software development can the tool represent. We only focus on information that is mentioned in the articles, and not on other important issues like if there is an underlying language, characteristics of the language, or if it is possible to represent relations between information. We select a subset of process elements [3] that we investigate if is present:
  - Activity – a process step, coupled to a human performer and to production tools.
  - Product – a software artifact that is produced.
  - Role – descriptions of rights and possibilities.
  - Human – process performers, undertakes roles, can be organised in teams.
  - Tool – used for software development, like a compiler, parser, CASE-tool.
  - Evolution support – support for static or dynamic variability of process models.
  - Project/organisation – consists of humans that have relations to each other.

* Enactment is the third topic of investigation. We examine if the tool offers any possibility to support software developers when they are actually developing software. With this we mean that a tool should be integrated with other tools for software development.

We will investigate if these features are present or not in the systems, or if it is not reported in the literature. We would also like to examine if the context is represented as a process model, and will use the definition in the previous section to differentiate process models from other types of models. We indicate if process models are used, other modeling techniques, just plain text, or if no context information is stored.

We finally indicate if the context information is used when retrieving new information – and computing the similarity between what is sought and what is present in a database. By “context information” we mean information that is not reused directly, but information that is used to identify if other information is suitable for a situation or not. One example might be that we have some experiences
on effort estimation which is stored in a system. By context information we mean information that explains which situations this knowledge is suitable for if it is going to be reused.

2 Results

Here we investigate a subset of papers from the knowledge management literature, which focuses on reuse of knowledge in the software engineering domain. The papers were selected according to the procedure described in the previous section. We first give a written summary of the use of process models in each of the selected papers, and then use the framework outlined in section 1.3 for further analysis.

2.1 Overview of tools

In the article [7], Bisio et al. present a system for cost estimation of software projects. The system is using Case-Based Reasoning to estimate the cost of a new software project based on analogy. The underlying assumption is that similar projects have similar costs. Software projects are represented with six attributes:

- Class of the product - network management system, administrative, etc.
- Size – code length, required documentation.
- Program characteristics – complexity, reliability.
- Developer group – skills, experience.
- Development environment – methodologies, tools, programming languages.
- User characteristics - stability of specifications.

The authors claim that this approach has added new insight into the field of effort estimation.

Grupe et. al. [8] have outlined a CBR system for defining requirements, estimating efforts, designing software, troubleshooting and maintenance. They have developed a case-base with descriptions of software projects, with the following information:

- Project name with textual description.
- Hardware environment for the project.
- Language or package the project was developed under.
- The size of the project in function points.
- A set of complexity factors like data communication, distributed functions, etc.

The authors see benefits mainly in software design consistency, cost- and effort-estimation and generally in streamlining software production.

Finnie et. al. [9] have made a prototype tool for estimating development effort based on analogy. They have compared the results of a CBR tool with regression testing, and a method using neural networks. CBR and neural networks gave “much better” results than regression testing, and CBR had a slight advantage in that it gives an explanation for the answers. However, no reference to what information was used for representing cases was given.
Aarts [10] is also using CBR for effort estimation which again uses an architecture, ProJort, for case and knowledge representation. Graphs are used for describing:

- Tasks – a well defined piece of work.
- Deliverables, which are the output of tasks.
- Resources that are allocated to tasks, and an estimate of the effort required.
- Projects – that are a set of tasks, and follows a particular process.
- Programs are “meta-project” that consists of all projects required to define a project. Programs also contain information of the hardware and operating system of a product.
- Expertise is divided in hardware, operating system, development environment, software layer, which also are subdivided.

The tasks are the cases in the system, but the other information is also used for matching cases.

Krampe et. al. [11] have developed a system that uses case-based reasoning for information system design. Specifications from former projects are reused, and a metalanguage is used for representing cases. This can be adopted to suit specific domains, but is not discussed in detail in the article. The article gives an example of how a sales system design can be modeled.

Mi et. al. [12] describe a unified resource model that is used for integrating characteristics of major types of objects appearing in software development models. The model again, has specialized models for: Software systems, Documents, Developers, Tools, Development processes.

Herlea et. al. [13] have applied a groupware knowledge management tool to manage the software engineering lifecycle, and place special emphasis on requirement engineering. The people involved in the requirement elicitation process are given different roles: A facilitator – organizes meetings, the users – who will forward the claims, an analyst who transfers requirements from the users to the design team, and finally the design team who will implement a system based on the requirements.

Different types of interaction are done in a system which has “virtual rooms” for various tasks such as storing “wish lists”, brainstorming, developing scenarios, record documentation and report information. These “rooms” are used both for storage, and for future retrieval.

### 2.2 Comparison of tools with process model technology

Having presented the tools, we now go on to use the framework outlined in section 1.3 to evaluate the tools.

| Table 1. Description of process elements used in the selected tools. |
In Table 1 we present which tools that make use of elicitation and enactment, and also how many process elements from the list in section 1.3 that are supported. The type of representation used for context in the tools are also stated in the table and whether context is used for reusing information or not.

## 3 Discussion

Now we go on to discuss the hypothesis that was made in the introductory section, and see what findings from the articles that describe the tools can influence information about the hypothesis described.

The overall question was “are process models suitable as a way to add context to experience that is captured for later use”. We then reformulated this in three hypotheses that were:

- What process model technology is used in the tools that exist now?
- How do these tools represent context?
- Is the context information used to identify information that is suitable for reuse?

We will now discuss each of these three hypotheses, and start with the first:

All the tools except one has some representation of a software product. Some use a textual description, like [8], whilst others classify products into groups as [7]. It is not surprising that this information is present in the tools, but why do people choose different representations? It seems that the tools that have a more narrow purpose, like estimation rather than to support the whole software development process have chosen a classification rather than textual description. An obvious reason for this is that it is easier to compare classes than to compare text strings, and thus it is easier to automate the search for reusable information with a classification. The system that does not supply information about the product [11], is concentrating on reuse of software design specifications, but it is somewhat strange that there is no relation to the final product that will be the result of the specification.

Other process elements that are used in the tools are activity, human, tool, and project, which are all used in three of the seven tools. Roles are used in two of them. It is interesting that none of the tools use the process element evolution.

From Table 1 we see that elicitation is possible in five of the seven tools, and seems to be well supported. Enactment, however is only supported in one.
If we move on to the second question to see how the tools represent context, we see from Table 2 that only one tool uses a representation language that can be described as a process modeling language. Five of the tools use just plain text for representation, and again some of them have different classes that attributes can have. One tool [10] uses a modeling language that can represent relations, but only has some of the process elements.

The last question concerns whether context information is used to identify information that is suitable for reuse. Five of the tools seem to make use of this context information to help with deciding if information is relevant or not.

4 Conclusion and Further Work

After the discussion in the previous section, we can conclude that:

- Elicitation is used in knowledge management tools for the software engineering domain.
- Information about process elements like products are widely used, information on activities, humans, tools and projects are used to some extent.
- Information about system evolution is not present in any of the tools examined.
- Context is mostly represented as text, and not with languages that are similar to process modeling languages.
- Context information is used to a great extent to identify if information is suitable for reuse.

So, are we able to say that process models are suitable as a way to add context to experience that is captured for later use? The investigation in the article gives some support to this claim, in that elicitation and some process elements are widely used, however the fact that only one of seven of the tools makes use of a process modeling language indicates either that this technology has been tried and found not to be suitable, or that process modeling technology is not very well known in the knowledge management community. However, if there had been experiments with process model technology in the knowledge management community, this should have been reported as it is just as good research to report failures rather than successes. So it seems likely that the reason for this that the process modeling and knowledge management communities have few intersections and have therefore focused on different aspects.

The final conclusion will therefore be that it is still an open question if process modeling technology is useful for reuse of knowledge. At least we can say that context information is important, and process models is one way of adding that.

Further work could be to actually try to make knowledge management tools reuse knowledge which is based on process models.
5 Acknowledgements

I would like to thank M. Letizia Jaccheri at the Department of Computer and Information Science for organising a PhD course in Process Modeling at NTNU where this paper was written, and also for providing feedback and help with articles. Further thanks to Zoran.Constantinescu-Fulop and Håvard D. Jørgensen at the same department for comments, and to Ralph Bergmann at the University of Kaiserslautern for help with finding relevant articles.

References