A Tool for the capture and use of Process knowledge in process tailoring

Peng Xu and Balasubramaniam Ramesh
pxu@cis.gsu.edu, bramesh@gsu.edu
Department of Computer Information Systems
Georgia State University

Abstract

Software processes are critical assets of software development organizations. The knowledge about context in which a process is defined and tailored is typically lost during software development activities, making the processes difficult to understand, reuse and evolve. We present a framework that represents the process knowledge used in defining and tailoring a software process Based on this model, we have developed a prototype tool to support the understanding, reuse and maintenance of this process knowledge.

1. Introduction

A software development process consists of a series of activities that are performed to transform users' requirements into a software system [1]. A well-defined and executed process helps create a software product with predictable quality meeting cost and schedule constraints [2]. As important assets of software development organizations [3, 4], well-defined software processes help improve the performance, predictability and reliability of software development work processes and increase productivity. They also make the software development approach scalable, transferable, measurable and independent on those assigned to work on different aspects of software development. Further, a well defined process also facilitates communications among team members and helps educate new project team members.

Creating processes from scratch rather than reusing them from prior experience is often inefficient and risky [5]. Reuse of software development processes increases the effectiveness of software development operations, and enables process engineers focus on enhancing and tailoring processes [6]. Often standard software development processes reference models such as ISO/IEC 12207 [7, 8], IEEE/IEA 12207 [9-11], and RUP® [12, 13] are used to develop organization and project specific software processes. Tailoring a software process model such as these may involve eliminating and/or adding elements, and changing workflows. In most organizations different processes that suit the needs of individual projects are in use simultaneously. In fact, it is not uncommon for a project to use different software development processes for different components of a product or even for different versions of the same component through its evolution. Though the software process standards such as ISO/IEC 12207 and IEEE/IEA 12207 require detailed documentation of the rationale behind tailoring a standard process, current practices on such documentation do not capture much of the critical information [14]. The objective of our research is to develop a framework characterizing "process knowledge" (which we define as the knowledge about the process of defining and tailoring a software process) and tools to support the understanding, reuse and maintenance of this process knowledge. Our work addresses the development and usage world modeling of processes (ie. the reasons for process engineering) proposed by Rolland et al [15].

First we discuss highlight the importance of our work with a discussion of related research. Then, we provide a brief description of our framework for representing process knowledge used in process tailoring. In the section 4, we discuss the architecture of a prototype system that supports process tailoring followed by conclusions.

2. Related Work

Recent research on process modeling recognizes that all processes are situational. I.e, software development is not a mechanical process where the same process can be used in every aspect of a project across all phases as well as across all projects. Software processes have to be adapted to meet the needs of specific circumstances. The need for such flexible approach to process definition is highlighted in Internet Software development activities where project teams use evolving

Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03)
0-7695-1874-5/03 $17.00 © 2002 IEEE

processes even across different versions of the same product [16]. McBreen [17] highlight the need to understand the fit between project circumstances and processes such as eXtreme programming before they can be successfully adopted. He emphasizes that "one size does not fit all" when it comes to software processes.

As adopting new software processes are very difficult in organizations, exisiting processes are retained as new processes are introduced. This may involve dropping, changing or adding new elements to a software process [17]. Method engineering, concerned with "design, constructing and adapting methods, techniques and tools for the development of information system." [18] also recognizes that methods need to be "situated" [19] - i.e., developed and evolved based on needs of projects and organizations [20]. For example, Punter [21] proposes an approach that can guide users in evaluating project characteristics and determining project strategies in constructing a method, highlighting the importance of learning how methods evolve with changing needs of a project. Agerfalk and Ahlgren [22] argue that in order to understand, learn, communicate, construct, and combining methods, rationale behind their specification must be recorded. Rossi et al [23] also view method rationale as a critical change and learning agent in an organizational method engineering practice. Method rationale can help keep track of changes in software methods, increase the fit into the given project or context, lead a new way to apply the method, and provide ways of communication and learning. Ralyte and Rolland .[24] proposed a meta-model for method fragment reuse based on capturing knowledge about the application domain and design activity. Similarly, KRAIEM et al.[25] proposed a meta-model for situational methods that "contingency factors" such as the that characterize a project and its environment. Brinkkemper et al. [26] proposed a framework to classify method fragments based on perspective, abstraction, and layer of granularity. They also proposed a set of formal rules for completeness and consistency check. The need for additional research on the role of factors affecting the definition and use of methods to suit specific situations, specifically organizational and project characteristics is identified in a recent survey of research in the field [27].

Recent studies on process tailoring activities [28, 29] [30] stress the difficulties involved in process tailoring such as the lack of precise guidelines in references models to support this activity. These studies also support our view that process definition and tailoring are knowledge intensive activities and that maintaining knowledge about the process definition and evolution will help in tailoring and reusing a process. Our approach is similar in spirit to research on method configuration. Investigating the adaptation of a particular method to various situated factors Karlsson et al. [31] proposed a method configuration process based on prototypical "Development Tracks and Generic Project Types".A Development Track is an ideal process configuration of standard methods suitable for a type of software development project. A Generic Project Type represents a set of recurrent project characteristics representing recurring patterns in organizations. When a project's characteristics match generic project types, the project can use the method configuration.

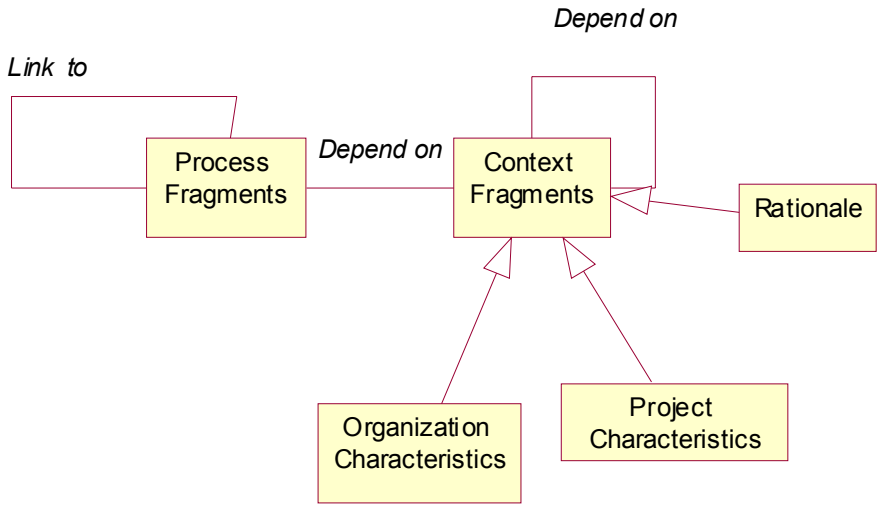

Figure 1. The framework

3. Framework for Process Knowledge in Process Tailoring

As a first step in supporting the process reuse, we have developed a framework to represent the elements needed to be captured in process tailoring. The framework is shown in Figure 1 and is described in detail elsewhere [32]. The elements in this framework include process fragments, products processed by process chunks, and different types of contextual knowledge that will help understand processes. This framework was derived from our empirical studies of problem solving behavior of experts involved in process tailoring in complex software development environments.

Process fragments are defined at at different levels of abstraction. For example, process fragments may be specialized into Phases, activities and tasks, say based on the RUP® model. Besides process fragments, context fragments are defined. Context fragments contains contextual knowledge under which a process is tailored. Organizational characteristics such as organizational policies and developer skills are factors that may affect process tailoring. Similarly, project characteristics such as application domains and stakeholders' concerns are represented in the model. These define the environment in which a process is specified and modified. Rationale behind process fragments is a critical element in our framework. This may be specialized into decisions, assumptions, intentions and alternatives considered in the choice of a specific process fragment in a given project. The organizational factors, project factors, and rationale are used in tailoring a process to suit the needs of an organization or project.

4. Tool for Capture and use of Process Knowledge

In this section, we discuss our prototype tool developed to capture and use process knowledte defined in terms of the contextual elements described in Figure 1 to support process tailoring. The tools is integrated with Rational Process Workbench®, an environment where process engineers can tailor the RUP® reference model for specific projects. The architecture of the prototype is shown in figure 2.

The prototype provides a tool for knowledge capture that is used to define and store process knowledge. Whereas the Contextual knowledge repository stores this knowledge, business process objects and process workflow objects are stored in a separate repository, such as the CASE tool's repository. A maintenance tool guides users in process tailoring. An inference engine maintains the consistency of the knowledge base of contextual knowledge and process models as they are incrementally defined and modified.

4.1 Acquiring Contextual knowledge

Process engineers, may define their process using a process construction tool such as Rational Process Workbench®. During this process, they can use our process knowledge capture tool. This tool provides an interface to define contextual knowledge fragments, relationships between different knowledge fragments, and relationships between knowledge fragments and process chunks. As mentioned earlier, a process chunk or fragment can be a phase, a workflow diagram, an activity, a task, a role, or an artifact. As a multi-user system with a web interface, this tool can be used by process engineers

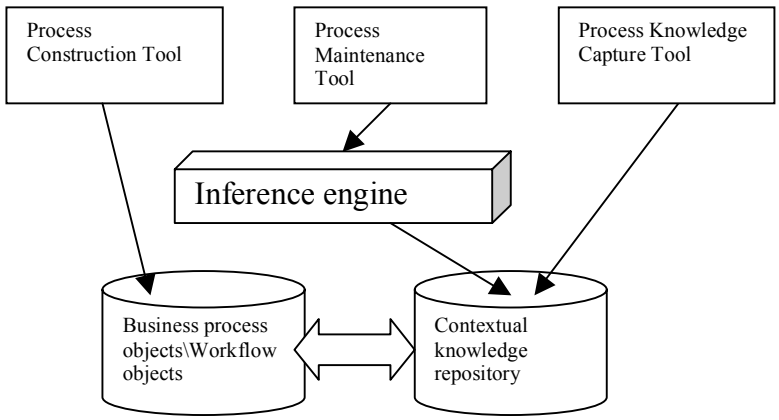

Figure 2. Architecture of Tool for Process Knowledge Capture and Use

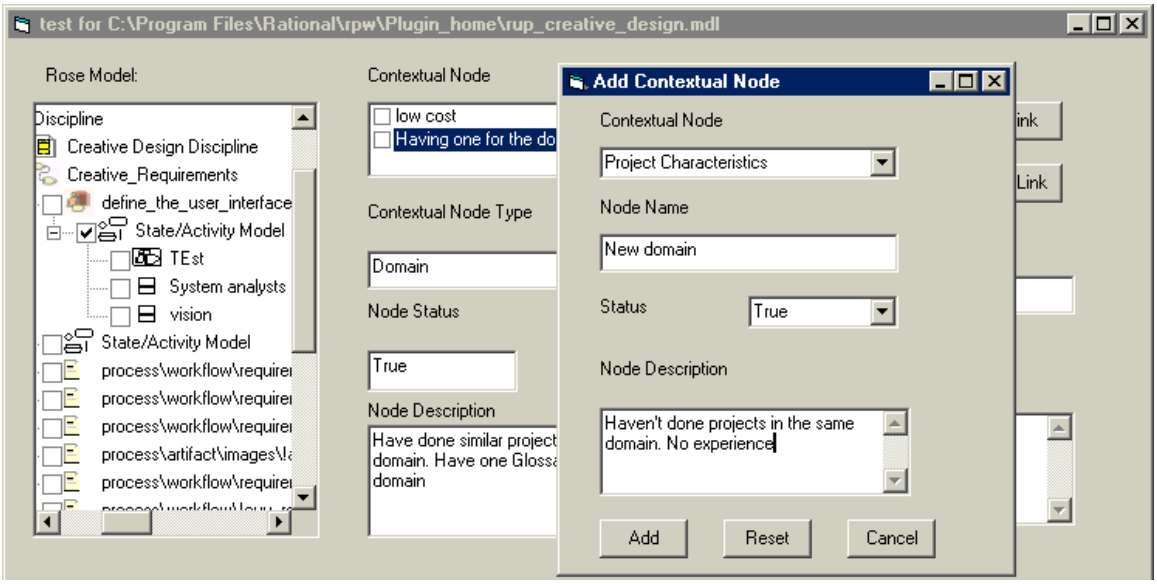

Figure 3: Process Knowledge Capture Tool

to discuss the definition and use of process elements. As these discussions proceed, knowledge about the organizational and project characteristics as well as the rationale behind the choice of specific process fragments is captured and linked to relevant process fragments. Our environment provides an inference enginet that is used to keep process knowledge consistent.

Figure 3 illustrates the use of our tool. Here, the RUP workbench has been used to define a process. Our tool extracts the elements of the RUP process model from the Rational repository and displays them as a tree in the left window. Process engineers can add and define the contextual nodes that are relate them to specific elements of the RUP (or tailored) model. Each process knowledge node may contain information such as node type, node name, status, and description. Figure 3 shows several process knowledge nodes including "Low cost", "Having one for the domain", and "New domain" as well the interface to acquire information about specific process knowledge fragments.

4.2 Process Knowledge Maintenance and Use

Software development organizations can develop one or more software process definitions by tailoring the industry or the international standards such as the RUP. The process knowledge captured using our tool helps developers understand the process chosen for a particular project / activity. With the help of the maintenance tool, developers can manipulate contextual knowledge. Any changes made to process knowledge fragments will be propagated through the various related elements. The affected fragments such as workflow activities or artifacts will be identified and alternative ways of achieving the proposed change will also be suggested.

Figure 4 shows a fragment of process knowledge defined using our tool. In this figure, three logic operators (And (&), Or (||), and Not (!)) are used to define the relationships between contextual knowledge and the decisions for process objects. Here, when deciding whether include the artifact "Glossary", the concerns of "low cost" and the project characteristic of "Having one for the domain" lead to the decision of "No Glossary". In contrast, when the project is new, the decision to use will be a "Glossary". The status of each process knowledge fragment representing whether it is true or false in the current situation is maintained within the system. When tailoring a process to a new project or situation, these status values can be evaluated to ensure that the current process is appropriate for the new context. If not, alternate configurations of process can be readily identified. In our example, if both if both "Low cost" and "Having one for the domain" are true, then decision of "No Glossary" will be made for a new project. However when these states change, the decision will have to be reevaluated based on the status of current environment.

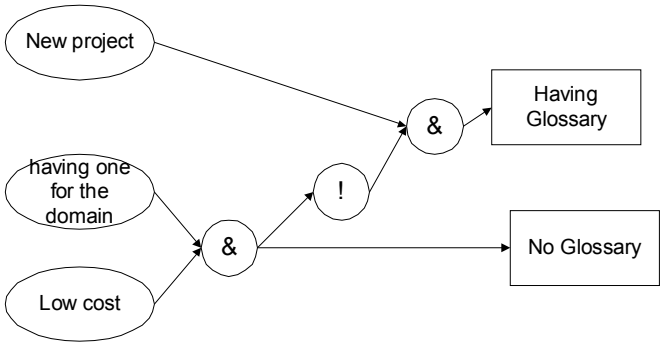

Figure 4. Fragment of process knowledge

Any change in contextual fragments may lead to addition, deletion, and modification of software process fragments. In a workflow model W= (N, L) where N represents nodes including process fragments and process fragments and L represents links between the nodes, the algorithm of propagating changes is described as the follows

a $\in N$;
a' = CHANGETYPE(a);
P $\subseteq N$, P={b | b is affected nodes in N when a is changed by CHANGETYPE () function};
While (P! = θ)
{
    b'=CHANGETYPE(b);
    Remove b from P;
    If (new node is affected) Add new node to P;
}

For example, the decision of not using "Glossary" artifact will lead to the deletion of the artifact from the process, and the deletion of activities/tasks that process this artifacts. The heuristic that propagate changes are defined as the following:

- Deletion
    - If the change leads to deletion of an artifact object, all tasks that process this artifact need to be updated. If the task only processes this artifact, it needs to be deleted. Otherwise, update the specification of the task. At the same, artifact set is updated.
    - If the change leads to deletion of an activity or a task, check if any artifact is created within this activity or task. If there is, then delete the artifact from artifact set, and update all that use such artifact. Check what other artifact are updated in the activity. Highlight such document object, indicating potential tailor of the document.
    - If the change leads to deletion of an event/condition, update such events/conditions.
- Addition:
    - If the change leads to a new artifact, update the artifact set.
    - If the change leads to a new activity or a task, add the activity/task and check input and output consistency.
- Modification:
    - If the change leads to modify an artifact, highlight all activities and tasks that process it.
    - If the change leads to modify an activity/task, highlight that activity/task and sub activities/tasks, and check the artifact consistency.

Thus our tool can be used to not only acquire, but also maintain a changing process knowledge base in the context of evolving project contexts.

5. Discussion

In summary, software development processes are valuable assets of organizations. Tailoring software process is a process involving intensive knowledge. This process knowledge is critical in helping developers to understand tailoring decisions and applying processes.Our research proposes a framework that provides guidelines on the elements of process knowledge that need to be captured when tailoring software processes. Further, a prototype system to facilitate the acquisition and use of this knowledge in process tailoring has also been developed. We are currently working empirical study to

evaluate the evaluate the effectiveness of the proposed approach.

References

1. Jacobson, I., G. Booch, and J. Rumbaugh, *The Unified Software Development Process*. 1999: Addison-Wesley Longman, Inc.
2. Pressman, R.S., *Software Engineering: A Practitioner's Approach*. 2001, New York: McGraw-Hill Companies, Inc.
3. Arbaoui, S. and F. Oquendo. *Reuse sensitive process models:Are process elements software assets too?* in *the 10th International Software Process Workshop*. 1996.
4. Greenwood, R.M., et al. *An Asset View on the Software Process*. in *the 10th International Software Process Workshop*. 1996.
5. Holdsworth, J., *Software Process Design: out of the tar pit*. 1998: McGraw-Hill International (UK) Limited.
6. Hitchings, R. and M. Martinez. *Reuse of Process Elements-One Company's Experience*. in *the 10th International Software Process Workshop (ISPW '96)*. 1996: IEEE.
7. Singh, R., *INTERNATIONAL STANDARD ISO/IEC 12207 SOFTWARE LIFE CYCLE PROCESSES*. 1998.
8. Singh, R., *An Introduction to International Standard ISO/IEC 12207*. 1999.
9. IEEE/EIA, *Industry implementation of International Standard ISO/IEC 12207.2: 1995. (ISO/IEC 12207 standard for information technology - software life cycle processes - implementation considerations.* IEEE/EIA, 1998.
10. IEEE/EIA, *Industry implementation of International Standard ISO/IEC 12207.0: 1995. (ISO/IEC 12207 standard for information technology - software life cycle processes - implementation considerations.* IEEE/EIA, 1998.
11. IEEE/EIA, *Industry implementation of International Standard ISO/IEC 12207.1: 1995. (ISO/IEC 12207 standard for information technology - software life cycle processes - implementation considerations.* IEEE/EIA, 1998.
12. Rational, R., *Rational Unified Process*. 2002.
13. Rational, *Rational Unified Process*. 1999.
14. Henninger, S. *Tools Supporting the Creation and Evolution of Software Development Knowledge*. in *the 1997 International Conference on Automated Software Engineering (ASE'97)*. 1997: IEEE.
15. Rolland, C. *A COMPREHENSIVE VIEW OF PROCESS ENGINEERING*. in *10th International Conference CAiSE'98,*. 1998.
16. Ramesh, B., R. Baskerville, and J. Pries-Heje, *Internet Software Engineering : A different class of Processes.* Annals of Software Engineering, 2002. 14((to appear)): p. 1-35.
17. McBreen, P., *Questioning Extreme Programming*. 2002, Reading, MA: Addison Wesley. 224.
18. Brinkkemper, S., *Method engineering: engineering of information systems development methods and tools.* Information and Software Technology, 1996. 38(4).
19. Hofstede, A.H.M.T. and T.F. Verhoef, *On the feasibility of situational method engineering.* Information Systems, 1999. 22(6/7).
20. Hillegersberg, J.V. and K. Kumar, *Using metamodeling to integrate object-oriented analysis, design and programming concepts.* Information Systems, 1999. 24(2).
21. Punter, T. and K. Lemmen, *The MEMA-model: towards a new approach for Method Engineering.* Information and Software Technology, 1996. 38.
22. Agerfalk, P.J. and K. Ahlgren. *Modeling the rationale of methods*. in *Information Resources Management Association International Conference*. 1999. PA, USA.
23. Rossi, M., et al. *Method Rationale in Method Engineering*. in *The 33rd Hawaii International Conference on System Sciences*. 2000.
24. Ralyte, J. and C. Rolland. *An approach for method reengineering*. in *ER*. 2001.
25. KRAIEM, N., I. BOURGUIBA, and S. SELMI. *Situational Method For Information System Project*. in *International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, and e-Medicine on the Internet*. 2000.
26. Brinkkemper, S., M. Saeki, and F. Harmsen. *Assembly Techniques for Method Engineering*. in *CAiSE'98*. 1998.
27. Arthur, H.M., t. Hofstede, and T.F. Verhoef, *On the feasibility of situational method engineering.* Information Systems, 1997. 22(6-7): p. 401-422.
28. Polo, M., et al. *MANTEMA: a Software Maintenance Methodology Based on the ISO/IEC 12207 Standard'*. in *Fourth IEEE International Symposium and Forum on Software Engineering Standards*. 1999.
29. Polo, M., et al. *MANTEMA: a Complete Rigorous Methodology for Supporting Maintenance based on The ISO/IEC 12207 Standard*. in *The Third European Conference on Software Maintenance and Reengineering*. 1999.

30. Demirors, O., et al. *Tailoring ISO/IEC 12207 for instructional software development*. in *The 26th Euromicro Conference*. 2000.
31. Karlsson, F., P.J. Agerfalk, and A. Hjalmarsson. *Method configuration with development tracks and generic project types*. in *The 6th CAiSE/IFIP8.1 International Workshop on Evaluation of Modeling Methods in System Analysis and Design*. 2001. Switzerland.
32. Xu, P. and B. Ramesh. *Towards Process Tailoring with Process Knowledge*. in *Workshop on Information Technologies (ICIS 2002)*. 2002. Barcelona, Spain.