Managing Software Process Knowledge

Ligia da Motta Silveira Borges
CDSV SA
Fernando Ferrari Av. 1000, CEP 29060-410
Vitória – ES - Brazil
ligias@zaz.com.br

Ricardo de Almeida Falbo
Computer Science Department,
Federal University of Espírito Santo
Fernando Ferrari Av., CEP 29060-900,
Vitória - ES - Brazil
falbo@inf.ufes.br

Abstract

In this paper, we discuss the crucial importance of storing and sharing the experience obtained in process definition to the continuous improvement of process quality. To share this knowledge, an experience repository should be built containing the organizational standard process as well as the artifacts and informal knowledge obtained throughout the projects. In order to facilitate the storage and sharing of the experience, we built a tool that supports the standard software process tailoring procedure for each project, providing a search mechanism.

1. Introduction

The demand in the software development area has increased and brought up some new requirements. Time to market, better quality and greater productivity are now critical aspects to the competitiveness of the software organizations. Besides this, the complexity of the software products to be developed has increased.

Since software quality is directly related to the quality of the process through which software is developed, one of the main directions pursued by researchers and practitioners is centered on the study and improvement of the software process [1].

To be effective and to lead to good quality products, a software process should be adequate to the application domain and to the specific project itself. Processes should be defined for each case, considering the application characteristics, the development team and the technology to be applied. Although different projects require processes with specific features in order to regard its peculiarities, it is possible to establish a set of software process assets for use in software process definition. This collection of software process assets is called standard software process. Projects tailor the organization's standard software process to develop their own defined software process, which accounts for the unique characteristics of the project. This tailored process is referred to as the project's defined software process [2].

Since the standard process tailoring for each project is a hard task, offering an automated support for this task becomes an important challenge.

The *Centro de Desenvolvimento de Sistemas de Vitória* - CDSV (System Development Center of Vitória) is inserted in this context. It is one of the five Xerox development centers around the world. CDSV has invested in the quality of its development process, having been certified as CMM (Capability Maturity Model) maturity level 3.

In 1998, after being certified as CMM-level-3, CDSV established an annual project, the SPI (Software Process Improvement) project, as a way for improving continuously the software process. As a result, goals are being established every year and these are used as basis for planning the improvement actions.

Based on previous SPI projects, the following issues were defined as being the most problematic ones in the CDSV improvement efforts:

- instantiation of the standard software process for specific projects;
- knowledge sharing during the projects; and
- use of project's feedback for improving the software process.

Updating the organizational software process is not enough. It is necessary to disseminate the lessons learned from the projects through the organization. An organization that does not register the successes or failures of its projects will have as a result the repetition of the failures. This organizational learning has not obtained good results at CDSV. Based on that, knowledge management (KM) was pointed as an interesting approach to deal with the main problems detected.

This paper discusses the use of KM in process improvement and presents ProKnowHow, a tool that was developed to support the standard process tailoring for the projects, allowing the knowledge acquired in this process to be shared. Section 2 gives an overview of KM and its

application by the software engineering community. Section 3 discusses the software development process and the difficulties found in its definition. Section 4 presents ProKnowHow, a KM-based tool for supporting project's software process definition from a standard software process. Section 5 discusses related works. The paper ends with a summary and a conclusion in Section 6.

2. Knowledge Management

Knowledge is considered to be a crucial resource of an organization, and then it should be carefully managed. Historically, organization knowledge has been stored on paper or in people's mind. Unfortunately, paper has limited accessibility and it is difficult to update [3]. Knowledge in people's mind is lost when individuals leave the company. Furthermore, in a large organization, it can be difficult to localize who knows some matter. So, knowledge has to be systematically collected, stored in a corporate memory, and shared across the organization.

Knowledge Management (KM) entails formally managing knowledge resources in order to facilitate creation, access and reuse of knowledge, typically using advanced technology. KM is formal in that knowledge is classified according to a prespecified ontology into structured databases [3].

The basic KM activities include: identification, capture, adaptation, integration, dissemination, use, and maintenance of knowledge. At the core of a KM system, it is a corporate or organizational memory, supporting reuse and sharing of organizational knowledge, including lessons learned [4].

Information technology plays a major supporting role in knowledge management [5]. A wide range of technologies are being used to implement KM systems, what includes databases and data warehouses, intranets and internet, browsers and search engines, intelligent agents and so on [3]. Also, several methods are being used to build corporate memories, such as knowledge engineering methods (e.g. CommonKADS), requirements analysis methods and ontology-based approaches [6].

Ontologies are particularly important for KM. They constitute the glue that binds KM activities together, allowing a content-oriented view of KM [5]. Ontologies define the shared vocabulary used in the KM system to facilitate communication, integration, search, storage and representation of knowledge [3].

In the software development context, knowledge reuse and sharing are a crucial asset for continuous improvement of the software process and consequently, the resulting products. The interaction between projects and corporate memory establishes two feedback loops. The first takes place during process execution, when knowledge obtained during the project course is analyzed and small changes to the execution of the process are applied (learning in project level). The second loop aims the knowledge packing at the end of the project, and the use of this knowledge in a new project, resulting in corporate learning [7].

The organizational goals determine the type of knowledge to be acquired by an organization. Any type of knowledge can be used to reduce rework and improve quality. Processes, quality models, development artifacts, expertise and lessons learned are examples of reusable knowledge. The proper storage of knowledge must, however, also be taken care of in order to achieve efficient reuse of knowledge in software organizations. A knowledge item generated in a project must, in general, be adapted to the need of future projects, and supplied with information capable to facilitate its reuse [7].

3. Software Process Improvement and KM

A software process can be seen as a group of activities, methods, tools and practices that are used to build a software product. Humphrey [8] defines software process as the group of necessary software engineering tasks to transform the users' requirements in software. In the definition of a software process the following information should be considered: activities to be accomplished, necessary resources, requested and produced artifacts, adopted procedures and the life cycle model used [9].

It should be noticed that there is no software process that can be applied to every kind of project, since every software development project is unique in some sense. Software type, application domain, team features, development technology and paradigm, project size and complexity, development methods, among other factors, have influence on the way a software product is acquired, developed, operated and maintained. However, there is a set of process assets, called standard process, that should be incorporated to any defined processes. The standard process defines a common structure to be followed by all projects, no matter which characteristics the software to be developed has. There is a number of reasons to define a standard process [8]:

- An organizational software process minimizes problems related to training, revisions and tool support;
- The experiences acquired in the projects can be incorporated to the standard process, contributing to improvements in all defined processes;
- Less time and effort are spent in defining projects' processes.

This tendency in using a standard process can be proved by several quality models and standards, such as ISO/IEC 12207, ISO/IEC TR 15504 and CMM. All of them suggest the use of a standard process as the starting point from which defined processes can be established. In this context, it is very important to promote continuous

process improvement and we should establish procedures for regularly updating the organizational software process.

But updating the organizational process is not enough. Lessons learned during a project's process definition should be shared with other project managers. In this context, an attractive approach is to offer automated support to software process definition, using KM. Such a tool needs to satisfy some requirements to allow developers communicate and collaborate, including:

- Its organizational memory should contain both formal and informal knowledge. Also, this organizational memory should be built based on an ontology defining a shared vocabulary;
- It should be defined the contents of the organizational memory. Also, a characterization scheme should be defined, specially to deal with informal knowledge retrieve and access.
- A systematic procedure for process definition should be establish in order to be automated;
- Politics for knowledge filtering should be defined. Because knowledge quality and importance varies from source to source, it is crucial to resort to knowledge filtering to ensure that the knowledge is really relevant;

All these requirements were considered in the development of ProKnowHow, a KM-based tool for supporting software process definition. It should be highlighted that ProKnowHow were defined in the context of CDSV and so it considers that there is a standard software process established. Thus a project's software process definition is in fact an instantiation (or tailoring) of the standard process.

4. ProKnowHow: A KM-based Tool for Supporting Software Process Instantiation

ProKnowHow was developed to achieve the following goals:

- To support the standard process tailoring for projects;
- To collect and disseminate the knowledge acquired during standard process instantiation;
- To support standard process updating based on the feedback from projects.

Thus, ProKnowHow had to satisfied the requirements set above. To deal with these requirements, ProKnowHow has the architecture shown in Figure 1.

The knowledge repository is divided in formal and informal knowledge repositories. The first is composed of process assets. The second stores lessons learned.

A project manager can use ProKnowHow to instantiate a software process for a project (Process Tailoring). In this task, he/she can search the knowledge repository to find relevant knowledge for his/her job (Knowledge Dissemination). Also during project's process definition the project manager can input informal knowledge about its job (Knowledge Capturing).

Finally, ProKnowHow offers a service for updating the standard software process, which is available for a Knowledge Manager. The Knowledge Manager is also responsible for adapting and approving a lesson learned input by a project manager (Knowledge Filtering).

Next, we present how ProKnowHow addresses the requirements pointed above.

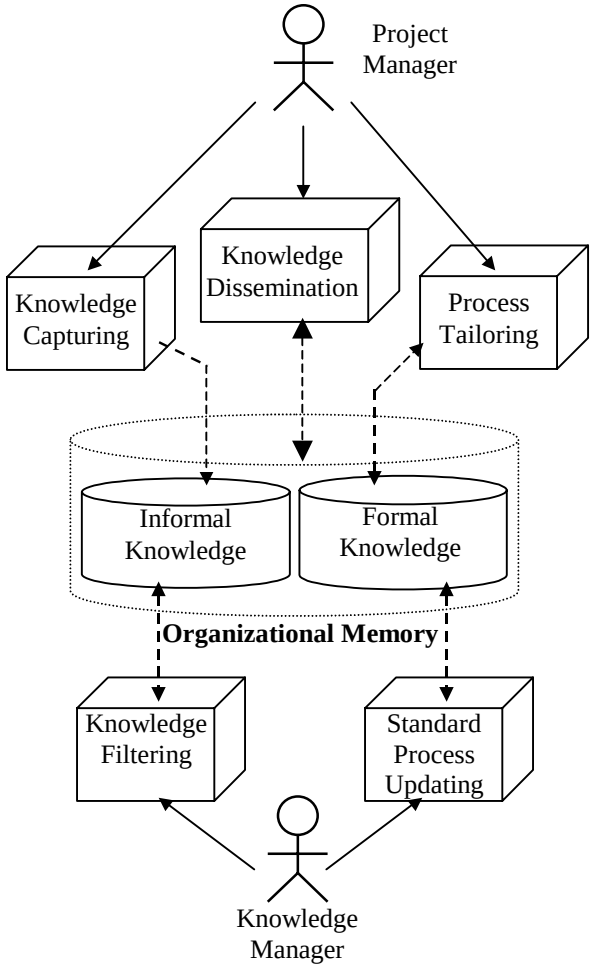

Figure 1. ProKnowHow architecture.

Organizational Memory (OM) Structure and Contents

First of all, the OM was built based on a software process ontology. This ontology was developed to support the acquisition, organization, reuse, and sharing of software process knowledge [9]. In ProKnowHow, it was used to establish a common vocabulary, facilitating process knowledge sharing and knowledge items search.

As pointed above, the OM should contain both formal and informal knowledge. In the process definition

context, there are two kinds of formal knowledge: software process assets and software process definition artifacts.

Once the standard process is the basis to the project's process instantiation, it is necessary to maintain its assets. Having the CMM as basis, the CDSV standard process is composed of the following assets:

- Life cycle models – a description of an ordered set of activities to realize the software development. Life cycle models are used as a reference in the definition of a software process, establishing macro-activities and the dependency relation between them. For each project, a life cycle model should be selected;
- Activities – tasks or pieces of work to be done during software development;
- Document models – patterns that define the format and guidelines for project artifact development;
- Procedures - well established and organized means for performing activities;
- Tools - software resources used to support the accomplishment of the activities;
- Policies - directions that govern the organization.

Artifacts are also formal knowledge. In the context of process definition, project's plan is the most important artifact produced. By using ProKnowHow, one can include, modify and exclude project's defined processes, using a configuration management system.

Lessons Learned are the informal knowledge handled by ProKnowHow. They are stored in the OM with the following information:

1. Project – indicates in which project the lesson was generated;
2. Process Asset – refers to process assets to which the lesson is associated;
3. Type of the lesson learned – identifies whether the lesson is positive (good practice) or negative (improvement opportunity);
4. Problem – a description of the problem being addressed;
5. Solution – a description of the solution to the problem;
6. Context – a description of the situation in which the lesson were generated.

Process Tailoring

ProKnowHow guides the project manager in the adaptation of the standard process for each project, suggesting life cycle models, activities, procedures, resources, and so on. Figure 2 shows an outline of the process tailoring procedure used in CDSV. It is composed of three main activities: project characterization, life cycle model selection and activity selection.

In the project characterization step, project characteristics are informed, as well as the project's desired quality profile. The main objective is to incorporate the project specific characteristics to the tailored process. The following set of information is of great importance:

- Project characteristics – include staff features, such as the user's ability to communicate requirements and team experience; problem features, such as problem complexity and application domain stability; product features, such as estimated product size and product type (off-the-shelf / customized solution); and development features, such as development paradigm and software type (Real Time Systems, Information Systems, Web Systems, and so on).
- Quality profile - in order to decide which activities of the standard process should be considered, a maturity level must be selected. Also, it is possible to personalize a maturity level by choosing which key process areas must be added;

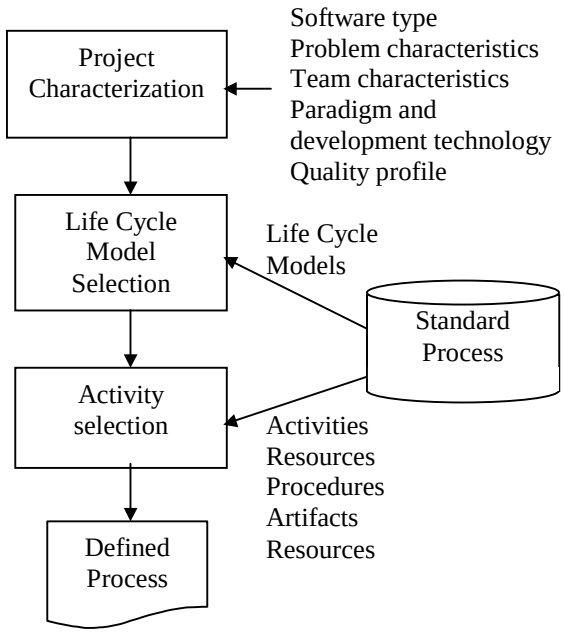

Figure 2. Standard process tailoring procedure.

Once the project is characterized, a life cycle model can be selected. Only life cycle models approved for use by the CDSV are considered in this step. Based on project's characteristics, ProKnowHow suggests life cycle models to be used, as shown in Figure 3. The project manager is free to accept or reject this suggestion.

Using the selected life cycle model, project's characteristics and the quality profile, a preliminary process can be proposed. In the activity selection step, the project manager can add or remove activities from the process. Also, for each activity, pre-activities, sub-activities, input and output artifacts, procedures, resources and tools should be defined.

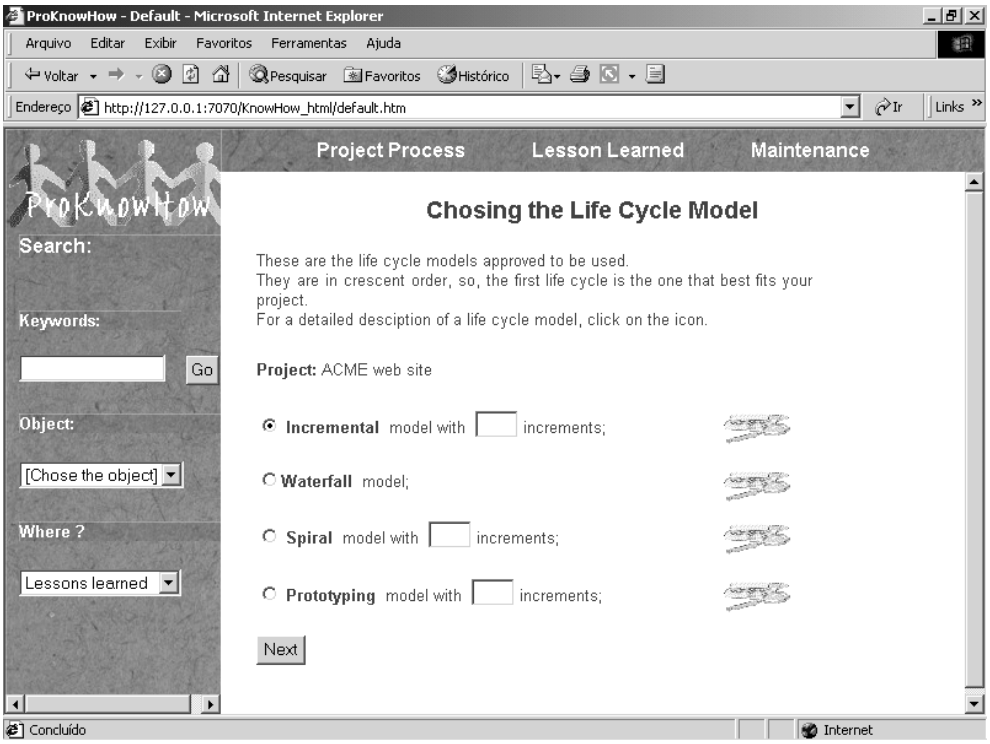

Figure 3. Selecting a life cycle model in ProKnowHow.

Management and quality control activities are selected according to the maturity level and key process areas chosen for the project. Management activities are those related to the project planning and tracking. Quality control activities, in turn, concerns those activities defined to assure the quality of the product being developed or the quality of the process used.

Development activities are directly related to the software construction process. They should be performed even if the project that does not have to reach the corresponding key process area. Therefore, their selection is closely related to the characteristics of the project as well as to the selected life cycle model.

Knowledge Capturing and Dissemination

Throughout the software process definition process, the project manager can ask for help. ProKnowHow offers a search functionality, shown in the left side of Figure 3. This reactive functionality can be used to retrieve both formal (process assets) and informal (lessons learned) knowledge. We can say that ProKnowHow also offers some kind of proactive behavior, since during process definition it always suggests software process assets according to the process definition step. In this case, only formal knowledge is considered.

The project manager is free to accept, or not, the suggestions given by the tool. However, if the resulting process does not conform to the standard process, he/she has to justify his/her attitude as a lesson learned. Also, the project manager can note comments about the guidelines that he/she has received from the tool. In this way, informal knowledge is captured. As shown in Figure 1, ProKnowHow also offers a functionality for maintaining the standard process, that is its formal knowledge.

Knowledge Filtering

When dealing with lessons learned, we have to consider another question. Project-level knowledge can be useful, but it is not always the case. Generally, project-level knowledge must be handled to become an organizational knowledge. In CDSV, the Knowledge Manager is responsible to check all lessons learned, and to decide if they should, or not, be available in the informal knowledge repository. Also, once defined that a lesson learned is really useful, the Knowledge Manager should make the appropriate changes to transform it in an organizational-level knowledge.

ProKnowHow supports a workflow for approving a lesson learned. First, a project manager inputs a lesson learned in the informal knowledge base. At this moment, this knowledge is not available for other project managers. The knowledge manager must to evaluate and adapt the lesson learned so that it can be considered an organizational knowledge. Once approved, the lesson learned is made available.

5. Related work

Several works have exploited the use of KM systems to support software engineering tasks.

Markulla [10] describes an initiative at ICL Finland to promote software engineering knowledge sharing and reuse. The focus is on supporting development tasks, such as planning, design and coding.

Althoff et al. [11] propose a generic, scalable architecture and its underlying methodology for reuse and continuous learning of all kinds of software engineering experience. Also they used a project as a scenario for demonstrating their approach. The focus is on reuse.

In [12], a system for supporting experience management in a multinational software improvement consultancy called Q-Labs is presented. The objective is to provide a "virtual office" for Q-Labs, and to allow Q-Labs consultant to benefit from the experience of every other Q-Labs consultants.

Looking to these works, we can find many common points. All of them, including our, are based on the concept of Experience Factory [13]. Lessons learned are used also in a similar way. However, none of them offers support for defining software processes from a standard software process. Thus, it is worth to remember that this work was developed in the context of a CMM level 3 organization and that the formal knowledge are process assets.

In [14], a tool called Assist-Pro was proposed to support software process definition according to the project specific characteristics. However, this work does not consider the use of a standard process as a basis for the process definition. Besides, Assist-Pro does not deal with informal knowledge.

6. Conclusions

This paper presented the CDSV's initiative to promote software process knowledge sharing, supported by ProKnowHow, a KM-based tool. ProKnowHow is now being implanted at CDSV's Intranet. We believe that it will contribute to process improvement at this organization, mainly because:

- Process definition task is being supported by a tool. Since ProKnowHow gives several advices, this task trends to become easier.
- ProKnowHow has potential for making the use of project's feedback in software process improvement easier. Since lessons learned are no more stored on paper, data can be processed in order to find improvement opportunities in the standard software process.

We expect to present actual results, corroborating our expectations or not, as soon as ProKnowHow's data are used by the CDSV's SPI Project.

Since CDSV is studying ways to achieve CMM level 4, we are now working to extend ProKnowHow to deal with software metrics knowledge. We have applied the Gol-Question-Metric model to define the CDSV's metrics set and we are now defining the contents of a metrics knowledge repository. Finally, we are also studying how agent technology can be used to improve ProKnowHow's proactive behavior.

References

[1] A. Fuggetta, "Software Process: A Roadmap", in Proc. of The Future of Software Engineering, ICSE'2000, Limerick, Ireland, 2000.

[2] M. C. Paulk, C. V. Weber, S. M. Garcia, M. B. Chrissis and M. Bush, "Key Practices of the Capability Maturity Model, Version 1.1", Technical Report CMU/SEI-93-TR-025, 1993.

[3] D. E. O'Leary, "Enterprise Knowledge Management", IEEE Computer, vol. 31, no. 3, pp. 54-61, March 1998.

[4] A. Abecker, A. Bernardi, K. Hinkelmann, O. Kuhn and M. Sintek, "Toward Technology for Organizational Memories", IEEE Intelligent systems, vol. , no. , pp. 40-48, May/June 1998.

[5] S. Staab, R. Studer, H.P. Schnurr and Y. Sure, "Knowledge Processes and Ontologies", IEEE Inteligent Systems, vol. , no. , pp. 26-34, January/February 2001.

[6] R. Dieng, O. Corby, A. Giboin and M. Ribière, "Methods and Tools for Corporate Knowledge Management", Int. Journal of Human-Computer Studies, vol. 51, no. 3, pp. 567-598, 1999.

[7] M. Broomé and P. Runeson, "Technical Requirements for the Implementation of an Experience Base", in Proc. of the 11th Int. Conference on Software Engineering and Knowledge Engineering , SEKE'99, Kaiserslautern, Germany, 1999.

[8] W.S. Humphrey, *Managing the Software Process*. Addison-Wesley Publishing, Company, Massachussets, 1990.

[9] R. A. Falbo, C.S. Menezes, and A.R.C. Rocha, "Using Ontologies to Improve Knowledge Integration in Software Engineering Environments", in Proceedings of SCI'98/ISAS'98, Orlando, USA, July, 1998.

[10] M. Markkula, "Knowledge Management in Software Engineering Projects", in Proc. of the 11th Int. Conference on Software Engineering and Knowledge Engineering, SEKE'99, Kaiserslautern, Germany, 1999.

[11] K-D. Althoff, A. Birk, S. Hartkopf, W. Muller, M. Nick, D. Surmann, and C. Tautz, "Managing Software Engineering Experience for Comprehensive Reuse", in Proc. of the 11th Int. Conference on Software Engineering and Knowledge Engineering, SEKE'99, Kaiserslautern, Germany, 1999.

[12] M.G. Mendonça Neto, V. Basili, C.B. Seaman, and Y-M Kim, "A Prototype Experience Management System for a Software Consulting Organization", in Proc. of the 13th Int. Conference on Software Engineering and Knowledge Engineering, SEKE'01, Buenos Aires, Argentina, 2001.

[13] V. Basili, G. Caldiera, and H. ROMBACH, "The Experience Factory", in: Encyclopedia of Software Engineering, vol..1, John Wiley & Sons, 1994.

[14] R.A. Falbo, C.S. Menezes, and A.R.C. Rocha, "Using Knowledge to Promote Knowledge Integration in Software Engineering Environments", in Proc. of the 11th Int. Conference on Software Engineering and Knowledge Engineering, SEKE'99, Kaiserslautern, Germany, 1999.