Using Knowledge Management to Improve Software Process Performance in a CMM Level 3 Organization

Ricardo de Almeida Falbo, Ligia S. Mota Borges, Fabio Feu Rosa Valente
Computer Science Department, Federal University of Espírito Santo, Vitória – ES, Brazil
falbo@inf.ufes.br, ligias@zaz.com.br, fabio.valente@terra.com.br

Abstract

Developing quality software products in the schedule and considering planned costs has always been a challenge to software organizations. The quality of a software product depends heavily on the quality of the software process used to develop it. This fact has led organizations to invest in improving their organizational software processes. In this context, Knowledge Management can be used to support process improvement. In this paper, we present the knowledge management approach adopted in a CMM level 3 organization to support organizational process tailoring to projects and process improvement based on metric data collected from past projects.

1. Introduction

Software quality is directly related to the quality of the process through which software is developed. Thus, one of the main directions pursued by researchers and practitioners is centered on studying and improving the software process [1]. In this context, software process definition is fundamental for achieving higher levels of maturity [2].

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 considering features of the application, development team and technology to be applied. But, although different projects require processes with specific features, it is possible to establish a set of software process assets for use in software process definition. This set of software process assets is called an organization's standard software process. The project's defined software process is developed by tailoring the organization's standard software process to fit the specific characteristics of the project [2]. However, tailoring the standard process for a project is not a simple task. It requires knowledge, experience and expertise from the project managers.

In this paper we discuss the knowledge management initiative carried out by the Xerox's System Development Center of Vitória (in Portuguese, Centro de Desenvolvimento de Sistemas de Vitória – CDSV). Its main goals are to support process definition and improvement based on metric data collected from past projects. Section 2 presents a brief report of CDSV process improvement efforts, and discusses some problems that motivated performing this initiative. Section 3 addresses how knowledge management (KM) can be used to improve software process performance. Section 4 presents ProKnowHow, a KM-based tool developed for supporting project's software process definition from a standard software process and improvement based on metric data collected from past projects. Section 5 discusses related works. Finally, section 6 presents the conclusions of our work.

2. Process Improvement Efforts in CDSV

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

A fundamental concept of software process quality in CMM is the organization's standard software process. An organization's standard software process is the operational definition of the basic process that guides the establishment of a common software process across the software projects in the organization. It describes the fundamental software process assets that each software project is expected to incorporate into its defined software process. It also describes the relationships between those software process assets, and establishes a consistent way of performing the software activities across the organization. It is essential for long-term stability and improvement [2].

The organization's standard software process is used to guide the definition of a project's software process that is the operational definition of the software process used by a project. The project's defined software process is a well-characterized and understood software process, described in terms of software standards, procedures, tools, and methods. It is developed by tailoring the organization's standard software process to fit the specific characteristics of the project [2].

At CMM level 3, one of the most important key process areas (KPA) is Organization Process Focus (OPF). The purpose of this KPA is to establish the organizational responsibility for software process activities that improve the organization's overall software process capability. It involves developing and maintaining an understanding of the organization's and projects' software processes and coordinating the activities to assess, develop, maintain, and improve these processes. The organization should provide the long-term commitments and resources to coordinate the development and maintenance of the software processes across current and future software projects via a group called software engineering process group (SEPG). This group is responsible for the organization's software process activities. It is specifically responsible for the development and maintenance of the organization's standard software process and related process assets (as described in the Organization Process Definition – OPD key process area), and it coordinates the process activities with the software projects [2].

To address this KPA, in 1998, after being certified as CMM-level-3, CDSV established an annual project, the Software Process Improvement (SPI) project, as a means for improving continuously its software processes. As a result, goals have been established every year, and used as basis for planning the improvement actions. Based on SPI projects from 1998 to 2001, the following problems were pointed out:

- In spite of having developed guidelines and criteria for the projects' tailoring of the organization's standard software process, in agreement with OPD, this activity was still hard to be done.
- As also defined in OPD, an organization's software process database should be established and maintained, to collect and make available data on the software processes and resulting software work products. A library of software process-related documentation should also be established and maintained. At CDSV, this database was developed as a set of documents, and a major problem arose: it was difficult to access them. Thus, knowledge sharing along the projects was not effectively happening.
- According to OPF, the strengths and weaknesses of the used software processes should be identified relative to the standard process. Information related to the use of the organization's standard software process by the software projects should be collected, reviewed, and made available [2]. This way, process improvement can be made based on the project's feedback. At CDSV, it also proved to be a hard task, especially when there are many projects.
- As pointed out in the Integrated Software Management KPA, the organization's software process database should be used for software planning and estimating. In other words, the database should be used as a source for estimating, planning, tracking, and replanning a software project. Data for similar software projects should be used when possible. CDSV's process database was not adequately design to support those tasks.

In sum, the most important findings were that it was necessary to share knowledge and disseminate the lessons learned from the projects through the organization. Knowledge is a crucial resource of an organization, and it should be carefully managed. CDSV was not able to manage knowledge and learn with the work done. Based on that, knowledge management (KM) was pointed as an interesting approach to deal with the problems detected.

3. Knowledge Management and Software Process Improvement

Knowledge Management (KM) entails formally managing knowledge resources in order to facilitate capturing, access, dissemination and reuse of knowledge, typically using advanced technology. KM is formal in that knowledge is classified according to a pre-specified ontology into structured databases [3]. In this sense, ontologies are particularly important for KM. They constitute the glue that binds KM activities together, allowing a content-oriented view of KM [4]. Ontologies define the shared vocabulary used in the KM system to facilitate communication, integration, search, storage and representation of knowledge [3].

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

In the software development context, KM can be viewed as the foundation 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 [5]. Using a KM approach, knowledge created during software processes can be captured, stored, disseminated, and reused, so that better quality and productivity can be achieved. KM can be used to better support several activities, such as software process definition, human resource allocation, estimation, requirement analysis, quality planning, and so on.

Analyzing the problems pointed out by the CDSV's SPI projects, we can clearly notice that KM is a promising approach to deal with them. Software development is a quickly changing, knowledge-intensive business involving many people working in different activities [6]. CDSV has problems identifying the content, location, and use of the knowledge. As pointed out by Rus and Lindvall [6], an improved use of this knowledge is the basic motivation and driver for KM in software engineering. Therefore, CDSV decided to invest efforts in it.

However KM is neither a product in itself, nor a solution that organizations can buy off-the-shelf. It is a process implemented over a period of time, which has much to do with human relationships as it does with business practices and information technology [7]. Moreover, as pointed out by Liebowitz [8], KM should start small and see what works. Thus, the KM efforts in CDSV were planned in order to establish a gradual competence in this area, including the following steps:

1. Initially, two main goals were establish: to develop a KM-based organizational software process database, and to provide KM-based support for tailoring the organizational process to projects.
2. Once established the database, the following step was to use its data for estimating, planning, tracking, and replanning software projects. The main goals of this step were to define metrics to be collected that could support better estimation, and to develop a KM-based estimation tool.
3. Finally, looking for leveraging CDSV to CMM level 4, the goal is to use metric data as basis for process improvement.

Currently, steps 1 and 2 are finishing, and as their main result, a tool called ProKnowHow was developed.

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

ProKnowHow is a KM-based tool for supporting software process improvement in CDSV. Its requirements include:

- ProKnowHow organizational memory should act as the organization's software process database. It should contain both formal and informal knowledge.
- The structure of the organizational memory should be well defined, and then ontologies have to be used to define it. Also, a characterization scheme should be defined, especially to deal with informal knowledge retrieve and access.
- Politics for knowledge filtering should be defined. Since knowledge relevance varies from situation to situation, knowledge filtering is essential to ensure that the knowledge retrieved is really relevant for the situation at hands.
- A systematic procedure for projects' process definition should be established in order to be supported by the tool.
- Software metrics should be defined based on the organization's objectives, and collected data should be properly presented to support estimation.

These requirements drove the main design decisions made during ProKnowHow development. First, we needed ontologies to ground the structure of the organizational memory (OM). Two integrated ontologies were selected for this purpose: the software process ontology, partially presented in [9], and the software metrics ontology, partially presented in [10]. Those ontologies were used to structure the OM, as well as to support knowledge items classification.

Second, we decided to apply the Quality Improvement Paradigm proposed by the Experience Factory (EF) [11]. This paradigm proposes an approach for software quality improvement that has many common aspects with KM, and that is focused on continuous quality improvement of software organizations. In fact, the EF Organization concept was before the term KM became popular, and the EF Organization addresses many of the same concerns [11]. The basis for the EF Organization concept is that software development projects can improve their performance by leveraging experience from previous projects [11]. The EF Organization separates responsibilities into two distinct organizations, as shown in Figure 1: the Project Organization, which uses past experiences to deliver new software products, and the EF, which supports software development by providing relevant experience. This organization applies directly to CDSV, since its SEPG acts as the EF, while the development area is the Project Organization.

Concerning software processes, 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 tailoring;
- To support standard process updating based on the feedback from projects.

To achieve these goals, the OM in Figure 1 must store knowledge about the various process assets, project's process plans, and lessons learned in tailoring the organizational process for projects. In the Project Organization of figure 1, *plan* means, in the case of software process, plan the process. *Do* means perform the project, following the project's defined process. In the EF, the *project support* should offer assistance for tailoring the organization's standard process to fit the project's characteristics. New projects' experiences should be analyzed and synthesized in order to support SEPG in updating and improving the organizational software process.

Concerning estimation and metrics, ProKnowHow has to meet the following goals:

- To support project estimation through retrieving data from past similar projects;
- To derive indicators from classes of projects performed by the organization;
- To allow relating software metrics to organizational goals.

To achieve these goals, the OM in Figure 1 must store knowledge about software metrics, project's estimation plans, including estimated and accomplished data, and lessons learned about estimation. In the Project Organization, *plan* means, in the case of estimation, plan the project, more precisely estimate the project. *Do* means track the project, comparing actual data to planned data. In the EF, the *project support* should support estimation. Finally, new projects' experiences should also be analyzed and synthesized.

Figure 2 shows ProKnowHow architecture and functionalities. Following, we discuss them in more details.

4.1. Organizational Memory

ProKnowHow's OM stores both formal and informal knowledge. Formal knowledge items can be artifacts produced during the software development or ontology instances. The latter is used to store general knowledge about the software engineering domain described through the software engineering ontologies that ground ProKnowHow: the software process ontology and the software metrics ontology, as previously cited.

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

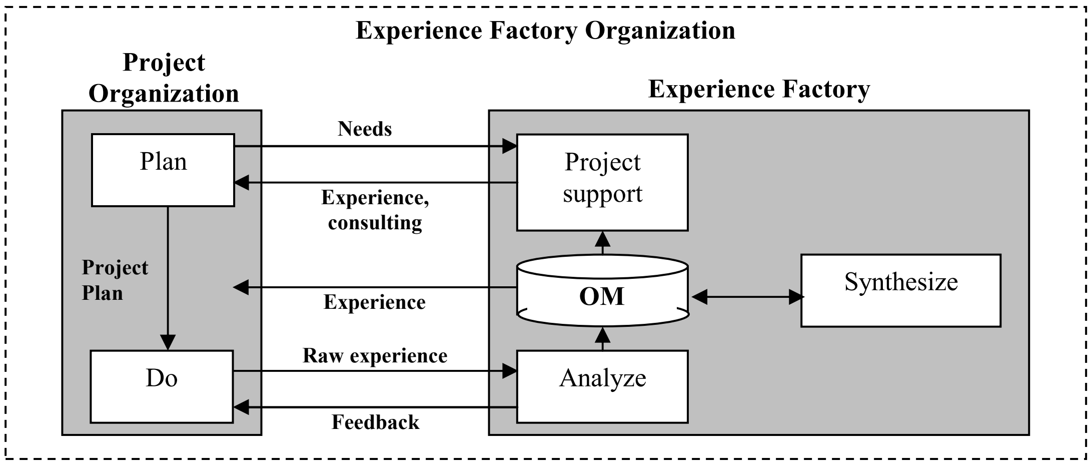

Figure 1. Experience Factory Organization.

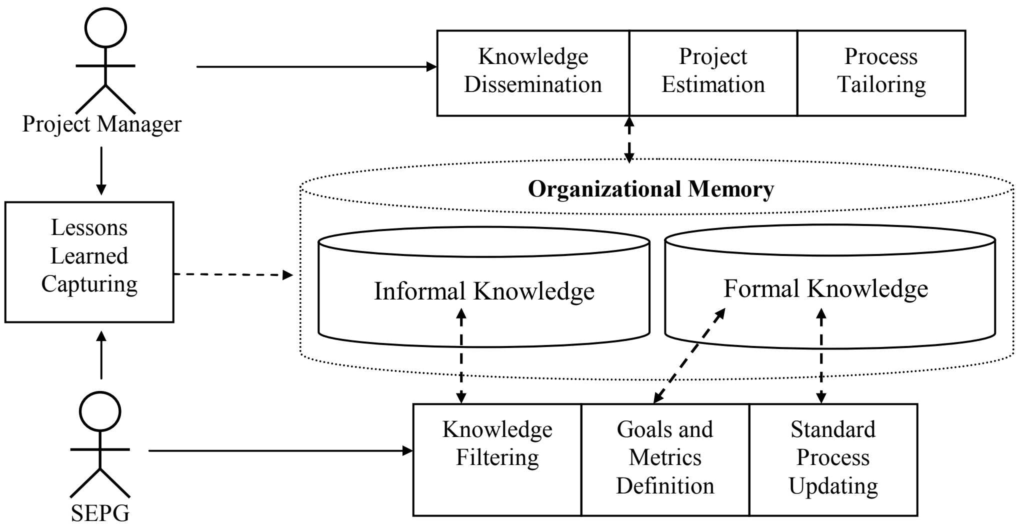

Figure 2. ProKnowHow Architecture and Functionalities.

- Project: indicates in which project the lesson was generated;
- Process Asset: refers to the process assets to which the lesson is associated;
- Type of the lesson learned: identifies whether the lesson is a good practice or an improvement opportunity;
- Problem: a description of the problem being addressed;
- Solution: a description of the solution adopted to solve the problem;
- Context: a description of the situation in which the lesson was generated.

As to process definition, the most important artifacts are the projects' software process plans. Ontology instances, in turn, concern describing software process assets. Once the standard process is the basis for the project's process tailoring, it is necessary to capture knowledge about its assets, including:

- Life cycle models – a description of an ordered set of activities to be used as a guide to the software process definition. They are used as a reference in a software process definition for a project, establishing macro-activities and the dependency relation between them;
- Activities – tasks to be done during software development;
- Artifacts – software artifacts that are produced and consumed by the activities;
- Resources – refer to people, tools, equipments, and so on, that are necessary to accomplish the activities;
- Procedures – well established and organized means for performing activities, including methods, techniques, and document models, which are patterns that define the format and guidelines for project artifact development;

Concerning estimation, the most important artifacts are the projects' plans. Since we have applied the GQ(I)M (Goal-Question-(Indicator)-Metric) paradigm [12], ontology instances describe knowledge about goals, quality characteristics (indicators) and metrics. In fact, the software metric ontology used does not address goals, as defined in GQ(I)M. But we decided to treat them as ontology instances.

4.2. Knowledge Management Services of ProKnowHow

As shown in figure 2, ProKnowHow offers a set of KM services that includes: knowledge capturing, retrieve and dissemination, and filtering. As discussed above, ProKnowHow's OM contains three types of knowledge items: artifacts, instances of ontologies and lessons learned. Then, ProKnowHow must offer facilities to capture each one of these knowledge types.

Artifacts created during the software process are submitted to configuration management and become available to ProKnowHow. Instances of ontologies are captured using the services for updating the standard software process (instances of the software process ontology), and for defining goals and metrics (instances of the software metrics ontology). Finally, there is a service for registering lessons learned.

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 (knowledge filtering). 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 lessons learned. First, a project manager inputs a lesson learned in the informal knowledge base. At this moment, this knowledge is not available for other developers. The knowledge manager must evaluate and adapt the lesson learned so that it can be considered an organizational knowledge. Once approved, the lesson learned is made available.

During software process definition or project estimation, the project manager can ask for help. ProKnowHow offers a search functionality. This reactive functionality can be used to retrieve both formal and informal knowledge. We can say that ProKnowHow also offers some kind of proactive behavior (dissemination), since during process definition it suggests software process assets according to the process definition step, and during project characterization it suggests metrics based on the established project's goals. In both cases, only ontology instances are 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 can be captured.

4.3. Organizational Software Process Tailoring and Project Estimation in ProKnowHow

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. The workflow 1 in Figure 3 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, including 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).

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. The project manager is free to accept or reject this suggestion.

Using the selected life cycle model and project's characteristics, a preliminary process is 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.

As previously cited, when selecting a life cycle model to the project or when selecting activities and its corresponding process assets, the project manager can use the retrieval service for searching past process plans or lessons learned. Also ProKnowHow proactively suggests software process assets according to the current process definition step, based on the part of the organizational memory that deals with the standard process.

ProKnowHow supports project estimation following the process shown in the workflow 2 of figure 3. The project characterization step, in this case, is a review of the project characteristics informed during process definition. In fact, there are some characteristics that are relevant only for estimation purposes, and thus were not treaded previously.

According to the GQ(I)M paradigm, there is a tight relationship between organization's goals and metrics, i.e. organizational metrics should be defined based on organizational goals. Project's goals, in turn, should agree with the organization's goals. This way, when an organizational goal is selected as a project goal, metrics for the project can be suggested, based on the organizational metrics for that goal. In other words, a project does not define its metrics. It defines its goals. Based on these goals, metrics are selected.

ProKnowHow supports defining goals and metrics for a project using an extension of this approach. First, the project manager should define the project's goals. During this step, he/she can use the retrieval service for searching similar past projects or lessons learned.

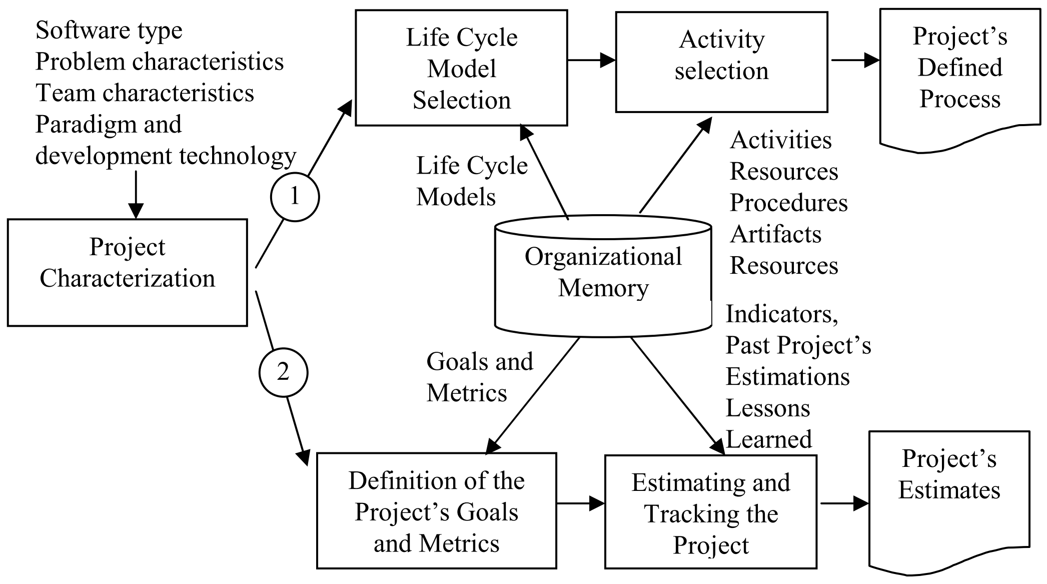

Figure 3. Estimating Projects with ProKnowHow.

Once defined the project's goals, quality characteristics (indicators) can be selected to treat those goals. As pointed out in the software metrics ontology [10], a software quality characteristic can be decomposed in others software quality characteristics. Thus quality characteristics and sub-characteristics should be defined to treat the goals. For each directly measurable quality characteristic, i.e. a software quality characteristic that is not composed of others software quality characteristics, a metric can be selected to measure it. For instance, suppose that the organizational goal "Improve software quality" is defined for a project and that the quality characteristic "Functionality" is selected as one of the indicators to manage that goal. Since functionality is not directly measurable, then sub-characteristics should be selected. If the sub-characteristic "suitability" is selected, the metric "functional adequacy" can be selected for measuring suitability. In this process, ProKnowHow proactively suggests quality characteristics and metrics based on the OM data defined early using the service for goals and metrics definition (see figure 2).

Once defined the project's metrics, the project manager can establish values to be achieved for those metrics in the project (project-specific goals). In the example above, a project-specific goal can be defined indicating that, in that project, functional adequacy should be greater than 85%.

Early detection and prediction of the quality of the software product is one of the most rewarding uses of metrics. Then, we should use the metrics selected in the previous step to estimate and track the project. When estimating the future values of the same metric by using past experience data, it is estimated based on a trend that is observed in a sufficient period of time. Thus, ProKnowHow focus on retrieving information from similar past projects in order to support project managers estimating and tracking their projects.

5. Related Work

Several works have exploited the use of KM to support software engineering tasks. For instance, the special issue of *IEEE Software* from May/June 2002 investigated KM's state of the practice in software engineering. In this issue, the articles report on the needs, issues, results, success factors, and lessons learned from a variety of KM applications [6].

Jay Liebowitz described a series of KM initiatives at NASA Goddard Space Flight Center [8]. Some of his conclusions also applied to the KM initiatives at CDSV, and reflect some of the strategies used. First, KM should start small. Several types of knowledge can be managed, but it is not possible to start thinking in all of then. CDSV opted by focusing in its processes and their improvement. Second, knowledge should be collected and disseminated during projects, when work is being done. ProKnowHow uses this approach, allowing project managers to search for knowledge when they are doing their work. Also, lessons learned can be registered during the project accomplishment.

Like ProKnowHow, several KM systems for software engineering are based on the Experience Factory concept. For instance, at Q-Labs [13], a system for supporting experience management was developed. The objective is to provide a "virtual office" for Q-Labs, and to allow Q-Labs' consultant to benefit from the experience of others consultants. At DaimlerChrysler [14], researches set up a Software Experience Center that reuses experiences from previous projects using a customized EF approach. However, none of these works offers support for defining software processes from a standard software process. Thus, it is worth to remember that CDSV is a CMM level 3 organization, and that some of its formal knowledge are process assets.

6. Conclusions and Future Work

This paper presented the CDSV's KM initiative supported by ProKnowHow, a KM-based tool. ProKnowHow was recently implanted at CDSV's Intranet, and we believe that it will contribute to process improvement, mainly because:

- Process definition task is now being supported by a tool. Since ProKnowHow gives several advices, this task trends to become easier, as it is being reported by project managers that started to use the tool.
- The use of project's feedback in software process improvement is becoming easier. Since lessons learned are no more stored on paper, it is being easier to use them in order to find improvement opportunities in the standard software process, as reported by SEPG.
- ProKnowHow has potential for making estimation easier, since past experience is being used to support estimating new projects.

We expect to present more results as soon as ProKnowHow's data are used by the CDSV's SPI Project. But we have already identified some problems. First, ProKnowHow search facility is not good enough, especially concerning finding similar projects. We are now working to improve this service using case-based reasoning. Second, proactive dissemination in ProKnowHow showed to be poor. We are studying ways to improve this service using agent technology.

7. Acknowledgments

This work was accomplished with the support of CNPq, an entity of the Brazilian Government reverted to scientific and technological development.

8. References

[1] A. Fuggetta, "Software Process: A Roadmap", in Proceedings 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] 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.

[5] 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.

[6] I. Rus, M. Lindvall, "Knowledge Management in Software Engineering", IEEE Software, pp. 26-38, May/June 2002.

[7] V.R. Benjamins, D. Fensel, A.G. Pérez, "Knowledge Management through Ontologies", Proc. of the 2nd International Conference on Practical Aspects of Knowledge Management (PAKM98), Switzerland, 1998.

[8] J. Liebowitz, "A Look at NASA Goddard Space Flight Center's Knowledge Management Initiatives", IEEE Software, pp. 40-42, May/June 2002.

[9] R.A. Falbo, C.S. Menezes, A.R.C. Rocha. "A Systematic Approach for Building Ontologies". Proceedings of the 6th Ibero-American Conference on Artificial Intelligence, Lisbon, Portugal, Lecture Notes in Computer Science, vol. 1484, 1998.

[10] R.A. Falbo, G. Guizzardi, G., K.C. Duarte, "An Ontological Approach to Domain Engineering", in Proc. of the 14th Int. Conference on Software Engineering and Knowledge Engineering, SEKE'02, Ischia, Italy, 2002.

[11] V.R. Basili, C. Seaman, "The Experience Factory Organization", IEEE Software, pp30-31, May/June 2002.

[12] R.E. Park, W.B. Goethert, W.A.Florac, *Goal Driven Software Measurement – a Guidebook*. Technical Report CMU/SEI-96-BH-002, Software Engineering Institute, Carnegie Mellon University, August 1996.

[13] 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, Buenos Aires, Argentina, 2001.

[14] K. Schneider, J-P. von Hunnius, V.R. Basili, "Experience in Implementing a Learning Software Organization", IEEE Software, pp46-49, May/June 2002.