Journal of Universal Computer Science, vol. 16, no. 3 (2010), 479-505
submitted: 4/10/09, accepted: 29/1/10, appeared: 1/2/10 © J.UCS

A Model for Capturing and Managing Software Engineering Knowledge and Experience

Gerardo Matturro
(Universidad ORT Uruguay, Montevideo, Uruguay
matturro@uni.ort.edu.uy)

Andrés Silva
(Universidad Politécnica de Madrid, Madrid, Spain
asilva@fi.upm.es)

Abstract: During software development projects there is always a particular working "product" that is generated but rarely managed: the knowledge and experience that team members acquire. This knowledge and experience, if conveniently managed, can be reused in future software projects and be the basis for process improvement initiatives. In this paper we present a model for managing the knowledge and experience team members acquire during software development projects in a non-disruptive way, by integrating its management into daily project activities. The purpose of the model is to identify and capture this knowledge and experience in order to derive lessons learned and proposals for best practices that enable an organization to preserve them for future use, and support software process improvement activities. The main contribution of the model is that it enables an organization to consider knowledge and experience management activities as an integral part of its software projects, instead of being considered, as it was until now, as a follow-up activity that is (infrequently) carried out after the end of the projects.

Keywords: Knowledge management, software engineering, experience capture
Categories: M.2, M.8

1 Introduction

IEEE Computer Society defines software engineering as the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software; that is, the application of engineering to software [IEEE, 90]. As commented by Ye, even though software development shares many characteristics with other engineering disciplines, it presents, however, new research challenges because it also strongly depends on the knowledge and the creativity of software development individuals [Ye, 06].

Knowledge management can be defined as the set of processes that govern the creation, transfer and utilization of knowledge [Gupta, 04]. Interventions into managing knowledge in organizations, also called KM initiatives, attempt at creating an environment that supports the handling of knowledge and ultimately lead to increased organizational effectiveness [Maier, 08].

Several authors have remarked the convenience to strengthening the links between the disciplines of software engineering and knowledge management as they consider that software engineering is characterized, precisely, by being an activity

intensive in knowledge. Kukko, Helander and Virtanen consider that the process of developing software is typically characterized as intensive in knowledge and that the outcome of the process, namely the software, is also a knowledge-intensive product [Kukko, 08]. Wohlin considers that the ability to manage knowledge in software engineering is a key success factor for software projects and also for software organizations in the future [Wohlin, 03]. Similar opinion has been expressed by Aurum, Daneshgar and Ward, when they mention that to keep software companies competitive in the market, developing effective ways of managing software knowledge is of interest to software developers [Aurum, 08].

One area within software engineering particularly in need of managing knowledge is the area of design and architectural knowledge. The discipline of Design Rationale aims to capture the knowledge and reasoning that justifies a resulting design. This includes how a design satisfies functional and quality requirements, why certain designs are selected over alternatives and what type of system behavior is expected, under different environmental conditions [Tang, 06]. As pointed out by Babar and colleagues, the availability of architectural knowledge improves the software development process and, if not properly managed, critical design knowledge will remain implicitly embedded in the architecture, becoming tacit knowledge that risks to be eroded as personnel change or leave [Babar, 09], [Kruchten, 05]. As Burge and Brown suggest, an explicit capture of the different design alternatives and their rationale is an activity that can be integrated into a design process model [Burge, 02].

Another important activity in current software engineering is the improvement of practices and development processes and, in this area, the perspective of knowledge management becomes particularly important. As Alagarsamy and colleagues mention, even if earlier works on software process improvement have considered a wide range of initiatives, knowledge management should be seen as a contemporary approach to refine the activities of software process improvement, since software processes have evolved from being processes carried away by data to being processes driven by information until the actual processes driven by knowledge [Alagarsamy, 07].

Briand considers that in software organizations, the knowledge and the experience acquired in previous projects can be used to improve the software practices in future ones, and that the organizational learning based on that experience is a key element in the effective adoption of new practices and for improving both productivity and quality [Briand, 03]. As Edwards mentions, even if the general literature on knowledge management contains many examples of successful knowledge management systems in use in companies related to information technology, few of those examples are specific to software engineering [Edwards, 03]. From his point of view, despite the fact that there is an active community of knowledge management in software engineering, their jobs are far from the mainstream on knowledge management. Bjørnson adds that software engineering has mainly dealt with the storage and retrieval of knowledge while topics such as the creation, transfer and application of knowledge need further attention [Bjørnson, 07].

Creation, transfer and utilization of knowledge are three knowledge management activities that necessarily happen in every software project, as also occur other activities considered as "traditional" such as requirement engineering, risk management, quality assurance and software testing, just to mention a few of them.

However, in our opinion, there are two aspects that differentiate these traditional activities from the knowledge management activities just mentioned: 1) the latter are taken into account neither in literature nor in practice because they are not explicitly planned or managed during the life cycle of projects, and 2) the activities of software engineering projects mentioned above as "traditional" generate products and artifacts whose scope are at the level of the project where they are created and managed, while products generated in the knowledge activities, that is, the knowledge and experience that project team members create and acquire, if properly managed, have a wider scope across different projects and even the entire organization.

According to Antunes, Seco and Gomes, the knowledge generated during software development projects is a potentially valuable asset for a software organization and, to exploit this asset, the organization must acquire, store and manage that knowledge for reuse [Antunes, 07]. And for this knowledge and experience to be able to be reused in future projects or in software process improvement initiatives, it first has to be captured and then disseminated and made available to the remainder of the organization as lessons learned and best practices.

Usual ways of capturing this knowledge and experience are mainly based on techniques such as semi-structured interviews ([Komi-Sirvio, 02], [Scott, 03]) or by applying methods such as project postmortem analysis ([Birk, 02]), post-project revisions ([Harrison, 03]), legacy sessions ([Cooper, 07]), and experience reports and reviews ([Dingsoyr, 01]).

The main drawback of these ways of capturing knowledge and experience is that the capture process is usually carried out after the occurrence of the experiences that are intended to be captured (usually after completion of the projects) and require that the holders of this experience, that means, the members of project teams, are available to participate in them, which is not possible in most cases ([Basili, 01], [Cooper, 07], [Desouza, 05]). If there is a time difference between the experience and its capture, there is the risk of that experience finally being lost. This risk is materialized if the owners of the experience are no longer available, either because they have been reallocated to another project or because they left the organization, taking with them the valuable knowledge and experience acquired.

Therefore, a new approach is required to enable organizations to capture and manage such knowledge and experiences as soon as they occur during the software projects life cycle and throughout. Thus, the problem we consider here can be stated as follows: “how to integrate to the software development practices and processes of an organization, procedures and specific artifacts to guide and manage the knowledge and experience that team members gain in software development projects”.

To answer this question, in this article we present an iterative model for managing the knowledge and experience that project team members acquired during their project activities, in order to identify and capture lessons learned and generate proposals of best practices that enable an organization to preserve this knowledge and experience, as well as to provide the basis to support an initiative to improve their practices and software processes. The proposed model, called "ele", has the advantage that its phases, activities and artifacts for managing knowledge, integrate to traditional activities of software projects and also support initiatives to improve the software practices and processes in use in an organization. The main contribution of this model is that it enables organizations to consider the activities of knowledge and experience

management as an integral part of the daily activities of their software projects instead of being, up to now, additional activities that are rarely carried out after the projects are completed.

As Pettersson mentions, although there has been an extensive research in the KM field, it is still complex and difficult for practitioners to implement KM in organizations [Pettersson, 09]. Considering this, the proposed model provides an organization with a starting point for managing its knowledge and experience in the software engineering field; gives specific directions on how to build a knowledge and experience repository, and provides the necessary mechanisms for ensuring that this knowledge and experience, in the form of lessons learned and best practices, will be used both in new software projects and in software process improvement initiatives.

The rest of the article is organized as follows. Section 2 describes existing approaches for capturing and managing software projects experience and also presents the main weaknesses of those approaches. In sections 3 and 4 we present our model and its artifacts for knowledge and experience management that support the creation, transfer and application of knowledge. In section 5 we show the way in which the phases of the model integrate to software project activities and also to the activities aimed to improve the software practices and processes in use in an organization. In section 6 we present the case study of an implementation of our model in a software organization. The purposes of this case study are (1) to show how the main flow of activities included in the model can be implemented in a software organization, and (2) to test its suitability for capturing and managing the knowledge and experience that project team members acquire during a software development project. Finally, in section 7 we present the conclusions of the article and future works.

2 Capturing and managing experience in software engineering

In this section we present a brief overview of existing approaches found in literature about capturing, preserving and sharing software engineering knowledge. The section concludes with a number of criticisms to the reviewed works, which constitute the starting point for designing our model.

2.1 Capturing software projects experience

As stated in section 1, usual ways for capturing the knowledge and experience from software projects are based on techniques such as semi-structured interviews ([Komi-Sirvio, 02], [Scott, 03]) or by applying methods such as project postmortem analysis ([Birk, 02]), post-project revisions ([Harrison, 03]), legacy sessions ([Cooper, 07]), and experience postmortem reports and reviews ([Dingsoyr, 01]).

A project post-mortem analysis, as described in [Birk, 02], comprises three phases: preparation, data collection, and analysis. The purposes of the preparation phase are to review all the documentation generated during the project in order to understand what has happened, and to determine the goals for the postmortem analysis. The data collection phase is the moment in which the relevant project experience is gathered and, once the important topics have been identified, they are prioritized before proceeding with the analysis phase. During this last phase, a

feedback session is conducted in order to analyze the data collected and to find the causes for positive and negative experiences.

Cooper has proposed an approach called legacy sessions, in reference to the working sessions where project team members identify innovations and improvements they have performed in their projects and that are potentially valuable for future users [Cooper, 07]. According to the description given by this author, a legacy session consists of four parts. The first part consists of a brainstorming session to identify potential legacies. These legacies represent apprenticeships that have the potential of being re-used by the members of the project team or by other members of the organization. In the second part, the participants synthesize the results of the former phase combining similar elements, taking out differential ones and categorizing the results as "processes", "products", "people" or other. From the final list, an element is chosen for subsequent discussion. The third phase is the detailed discussion of the chosen element for, in the fourth phase, create a summary of the discussion, following a predefined structured pattern.

Harrison presents the post-project reviews as a way of providing a formal mechanism to transfer experience from a project team to an organizational memory once the project has finished and while these experiences are still fresh in the minds of the participants [Harrison, 03]. The captured experience is stored in a repository of learned lessons whose purpose is to facilitate the organization, maintenance and spread of the captured knowledge. The repository is based on web technology and it has an interface based on filling-in-forms for the people who provide learned lessons can add new experiences to it.

Dingsoyr, Moe and Nytro mention other two methods of capturing experiences from projects: the experience reports and the post-mortem revision [Dyngsoyr, 01]. An experience report is a predefined structured document whose aim is to get information from the participants, make them discuss about the way in which the project was managed and finally analyze the causes of why things succeeded, or not, throughout the project. For post mortem revisions, the authors mention the use of techniques such as the KJ method for brainstorming and Ishikawa diagrams for analyzing causes of those relevant aspects that were identified with the former technique. The KJ method (after its author, Kawakita Jiro) or "affinity diagram" is used to refine a brainstorm into something that makes sense and can be dealt with more easily.

2.2 The Experience Factory approach

More comprehensive approaches on capturing and managing software engineering knowledge and experience for reusing them in future projects can be traced back to the approach known as Experience Factory, proposed by Basili, Caldeira and Rombach [Basili, 94].

As defined by its authors, an experience factory is a physical and/or logical infrastructure that supports development projects, analyzing and synthesizing experiences of all kinds, acting as a repository of such expertise and providing, on demand, that experience to other software development projects. This approach divides the efforts of software development into two separate units with responsibility for developing software projects and for capturing experiences. The Experience Factory unit is responsible for developing, updating and providing reusable

experiences to software development teams. The artifacts of experience can be generated on demand for any units of software development (known as Project Organization) or by an independent analysis of information obtained to existing projects. The physical implementation of an experience factory is an Experience Management System, consisting of content, structure, processes and tools [Basili, 01]. The content (which may be data, information, knowledge or experience) and the structure (that is the way in which the content is organized) constitute what is called an Experience Base. The procedures are instructions on how to handle the experience base and the tools supports the content management and the implementation of procedures. An effective experience base contains an accessible and integrated set of experience models, analyzed, synthesized and packaged, that capture previous experiences. Jedlitschka and colleagues comment that the approach of Experience Factory includes the collection, documentation, dissemination and storage of experience (in the form of "experience packages") in an experience base, which is the organizational memory for the knowledge and experience that is relevant for the organization [Jedlitschka, 01].

From this seminal work, Schneider, von Hunnius and Basili ([Schneider, 02]) report the implementation of a project at DaimlerChrysler called Software Experience Center (SEC), based on the concept of Experience Factory commented above. The operational objective of the SEC was to provide business units with the concepts of a learning organization and a prototype of an experience base. The SEC supported all activities, from experience elicitation to making available these experiences for the software task at hand.

Johansson, Hall and Coquard ([Johansson, 99]) report on the implementation at Ericsson AB Software Technology of a variant of the Factory Experience called Experience Engine. This variant, unlike the Experience Factory, which is based on experiences stored in experience bases, is based on tacit knowledge. Koskinen defines tacit knowledge as the knowledge that is acquired through practice and experience [Koskinen, 01]. To make the tacit knowledge accessible to a wider group of people, two new roles have been defined in the organization. These are the “experience communicator” and “experience broker”. The former is a person who possesses a deep knowledge about one or more topics, while the experience broker has the mission to connect the communicator with the person or people who have a problem. The role of the communicator is not to solve the problem, but to educate and assist the possessor of it about how to fix it.

Komi-Sirvio, Mantyniemi and Seppänen ([Komi-Sirvio, 02]) have proposed a slightly different approach to the Experience Factory, in which knowledge gathered from past projects is guided by the immediate and specific needs of an ongoing project. This approach consists of a "knowledge capturing project” and "customer projects." The knowledge capturing project collects knowledge from relevant sources, "package it" and provides it to a new customer project for reusing it on demand in a similar way as the Experience Factory operates. This solution, as mentioned by its authors, does not modify the organizational settings and does not require new tools. Knowledge comes from existing sources such as final projects reports, errors databases, discussion forums and, what is the most important, people. The process of knowledge capture uses as the main technique the semi-structured interview.

2.3 Criticisms to the reviewed works

With regard to the project post-mortem analysis and similar techniques, all of them are characterized by the fact that the capture of the experience is done later in time, with regard to the actual occurrence of the experience, generally after finishing the project or at least by the time it has reached a relevant milestone.

Several criticisms have been formulated in relation with this fact. Basili, Lindvall and Costa mention that the problem with these post-mortems is that they arrive very late in the life of a project, if they are ever done [Basili, 01]. Desouza, Dingsøyr and Awazu consider as an unfortunate situation the fact that most of the software engineers do not have time to finish a project because they are almost immediately assigned another one [Desouza, 05]. Cooper mentions precisely the fact that project teams are by definition, temporal entities and that once the project is finished the members return to the organization taking with them the individual knowledge [Cooper, 07]. Zedtwitz mentions the restrictions and the lack of time as one of the main reasons for skipping the post-mortem revision, as organizations have a queue of projects that project managers and other team members have to be assigned as soon as they finish the current one [Zedtwitz, 02]. Keegan and Turner conducted a study in 19 companies with practices in place (referred as after action reviews and project end reviews) aimed to capture the learning that takes place after projects are completed [Keegan, 01]. In their findings, the authors pointed out that, frequently, project team members do not have the time for meetings or for lessons learned review sessions. Often, project team members are immediately reassigned to new projects before they have the time for participating in lessons learned sessions or after-action reviews.

In consequence, to separate in time the experience and its subsequent capture involves some risks. Experience might be lost if the people in charge of the project are no longer available because they have been assigned another project or even because they have abandoned the organization, taking with them valuable knowledge and experience acquired, sometimes at high cost.

With regard to the experience factory and related approaches, there are also several criticisms found in the literature. As Chau and Maurer pointed out, the Experience Factory framework sets which are the activities of knowledge management that are needed (elicitation, analysis, generalization, packaging and dissemination) but presents the lack of not giving directions on how such activities should be carried out [Chau, 05]. Similar opinion has been exppresssed by Zhu, Staples and Gorton, who believe that one of the problems in the current investigation in experience repositories is that most of it focuses on the technological aspect for building repositories rather than on how really occur the capture and share of experiences [Zhu, 07].

Another criticism that this approach has received concerns the access to the "experience packages" provided through the repository of experience are maintained by a group dedicated to the task of generalizing these experiences as much as possible for its reuse, implying that to feed or update the contents of the repository it is necessary to go through a controlled, and usually slow, process [Chau, 05].

Finally, two additional critiques can be added to the previous ones. On the one hand, the experience factory approach has the premise of capturing experiences "of all kinds" but without first identifying which are the software practices or processes to be improved, or what kind of experiences are of interest to the organization, according to

their current or future needs. On the other hand, and particularly in the approach of experience factory, the authors suggest the use of software tools such as Frequently Asked Questions, e-mail and chat sessions to capture and exchange of experiences [Basili, 94], but they don't establish guidelines on how to make an efficient use of these tools for such purposes.

3 The Model "ele"

Based on the criticisms presented above, we started to think that a different approach was needed, focusing on a) improving the process of capturing project experience, avoiding the weaknesses of the post-mortem analysis and similar approaches regarding the moment in the project life cycles they are usually applied, and b) emphasizing the knowledge management processes and related tools, and not exclusively on the repository, even though the existence of such a repository continues to be an essential element, both for preserving the captured knowledge and as a vehicle for knowledge and experience dissemination.

Our model, presented in the next sub-sections, places its emphasis on the process for capturing knowledge and experience, delimiting the kinds of knowledge and experiences that are worth capturing according to the organizational needs, by making use of a specific tool, called "reflective guide", designed for this purpose.

3.1 General Overview of the Model

We start by showing, in Figure 1, the flow diagram of the main activities of our proposed model, from the initial definition of the objectives of knowledge and experiences that are intended to capture, until its end, when the lessons learned and best practices are identified, elaborated and integrated into a repository, for making them available to the rest of the organization for its use in new software projects.

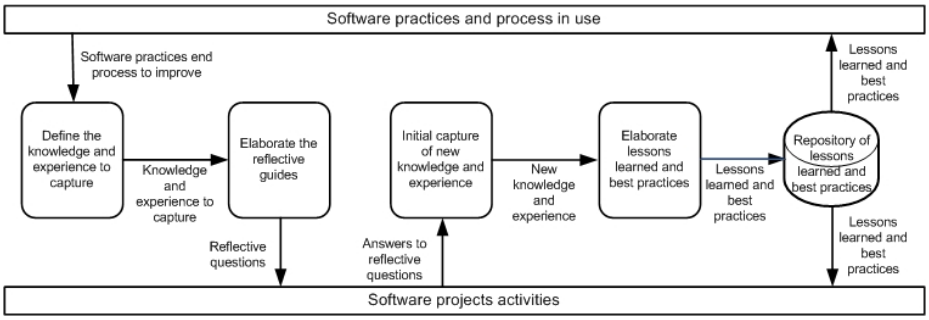

Figure 1: General overview of the Model "ele"

Consider that there is a software organization that aims to improve its software practices and processes, based on the management of the knowledge and experience acquired by the participation of its members in different software projects. We understand by "organization" a company or a business unit within a company whose

core business is to develop software and which carries out activities to improve its software practices and processes.

From the set of practices and software processes in use in the organization, we define a series of objectives for new knowledge creation for those software development practices and processes that the organization considers necessary to improve. Based on these objectives the so-called "reflective guides" are developed, defined as a knowledge management tool whose purpose is to guide the reflection of the project team members on those aspects of project activities for which there is a need to capture experiences. These guides contain a series of questions or sentences whose purpose is to provide team members with an opportunity for a primary registration of the knowledge and experience acquired. These questions and sentences remain the same during the iteration of the model but, as we explain in sub-section 3.2, the model provides an opportunity to evaluate their usefulness for promoting reflection, and to change them, if needed, to best suite this purpose.

Asking questions is nothing new. The difference in our approach is that these questions are asked "before" the team members perform their project activities and not "after" (and in general, long after) those activities have been performed. In this way, respondents know in advance the questions he/she will have to answer later, and find them in a better position to reflect on and to give a more detailed answer.

Once these guides are elaborated, they are assigned to the software project team members who, during the time in which they execute project tasks, use these guides to register their reflections and impressions, difficulties found, unforeseen facts and similar considerations related to the manner in which they carry out his/her tasks.

When the project tasks that the reflective guides relate to are completed, and after the reflective questions or sentences have been answered, those answers are compiled for a primary analysis and for the initial identification of new knowledge and experiences captured in the answers.

This last activity provides new knowledge and personal experiences that serve as an input to the subsequent process of collective analysis and discussion aimed to identify the lessons learned and proposals of best practices that, later, will be incorporated into the Repository of Lessons Learned and Best Practices. These knowledge and experience will impact on how project activities will be carried out in the future, and be the basis for future improvements to the software practices and processes used by the organization.

This sequence of activities is repeated iteratively in order to enable the organization to manage, in an incremental manner, the creation of knowledge. Once new lessons learned and proposals of best practices have been obtained in one iteration, a new iteration can begin to test them in the next software project. In this way, it is possible to gather more insights about they appropriateness and to allow the organization to continuously refine the knowledge and experience already captured. A new iteration will also enable the organization to define new objectives for experience capture from a different set of practices and process. In this case, new reflective guides will be elaborated, and the cycle continues as described above.

Iterations also enable organizational learning by integrating the activities of knowledge and experience management into software projects and into the activities aimed to improve software practices and processes.

3.2 Structure of the Model ele

We are now in position to provide more details about the phases of the model "ele", as shown in Figure 2.

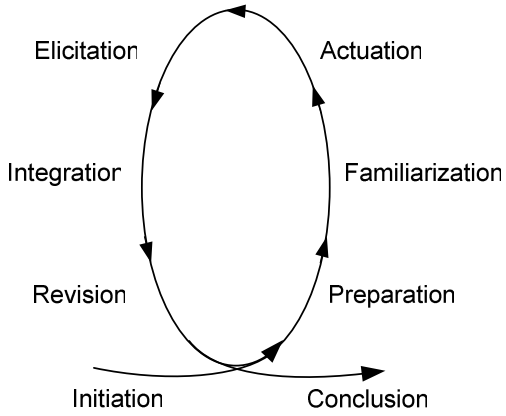

Figure 2: The model "ele"

The name given to the model comes after the shape of the letter "l", chosen for its graphical representation, and pronounced as "eh-leh" in Spanish.

The model begins with an **Initiation** phase, and its purpose is to establish the groundwork for launching the model. Here is where the practices and activities of software engineering on which the model will be applied are defined (those that the organization aims to improve), the setting out of objectives of new knowledge and experiences that are intended to capture, and defines the roles and the responsibilities of different actors who will participate in implementing the next phases of the model.

This initial phase is followed by an iterative cycle consisting of stages from Preparation to Revision, as described below.

The **Preparation** phase aims to prepare the organization for a new iteration of the model. Here, the above-mentioned Reflective Guides are elaborated and updated according to the objectives of knowledge and experience capture, previously defined.

The purpose of the **Familiarization** phase is to deliver to project team members the reflective guides related to the tasks they are carrying out, to make sure they know and understand the established objectives of capturing knowledge and experience. Its purpose, additionally, is to inform team members about the lessons learned and best practices related to the project activities they are carrying out, which have been identified and developed in previous iterations of the model.

In the **Actuation** phase, the members of the project teams, in parallel to their assigned project activities, will analyze them based on the criteria provided in the reflective guides, by answering the corresponding questions. Also in this phase the members of the project teams begin to delineate proposals of improvements that will be analyzed and discussed later in the Elicitation phase.

For the **Elicitation** phase, the intention is to analyze the answers gathered in the reflective guides and to synthesize the knowledge and the experiences acquired, with

the aim of identifying and extract lessons learned from the practical execution of project tasks, and to generate proposals of best practices for the items that were selected during the Initiation phase. This phase is implemented by conducting an experience elicitation workshop, as explained in subsection 4.2 below.

The **Integration** phase aims to incorporate, into the repository of lessons learned and best practices, the knowledge and experience captured in the previous phase. This integration includes the incorporation of new elements of knowledge and experience into the Repository, as well as the reformulation of the existing elements, in order to update the lessons learned and the best practices. The structure of the repository and the way it is intended to be used are explained in subsection 4.3 below.

The last phase of the iterative cycle, **Revision**, aims to evaluate the application of the model so far, in order to identify eventual deviations from the proposed goals related to knowledge and experience capture, as well as to identify areas for improving the implementation and execution of the model itself.

Finally, the **Conclusion** phase aims to formally close the iteration for the practices and processes established at the beginning. This stage is reached when, according to the organization, new or improved lessons learned and proposals of best practices have been obtained for the practices selected in the Initiation phase.

4 Other Elements of the Model

The proposed model is complemented by a set of artifacts that support knowledge and experience management activities, used in its different phases. These artifacts, described below, are the already mentioned Reflective Guides, the Experience Elicitation Workshop and the Repository of Lessons Learned and Best Practices.

4.1 The Reflective Guides

Central to the model are the reflective guides, as they are the main tool for knowledge and experience capture. These guides, as we propose them, constitute a special type of reflective journal that records a learning item that took place as a result of reflecting on experiences and situations, and include a series of questions and statements whose goal is to motivate and facilitate personal reflection activities, by directing the attention of team members to those aspects of the tasks for which knowledge and experience capture were initially defined.

To define the types of reflective questions or sentences, we propose to use, as a framework, Bloom's taxonomy of educational objectives for the cognitive domain [Bloom, 56]. This taxonomy, as initially defined by its authors, refers to a classification of the different objectives that educators set for students. However, other uses are possible and it has been used, for example, in the Guide to the Software Engineering Body of Knowledge (SWEBOK) as an aid for defining course material, job descriptions, professional training programs or professional development paths [Abran, 04]. Similar uses of Bloom's taxonomy can be found in [Azuma, 03] and in [Bourque, 03]. Based on the six levels of this taxonomy, we can formulate questions or sentences that activate different cognitive operations, from the simple recall of facts up to the more complex processes of synthesis and evaluation of information. The questions or statements to include in the guide must refer to the software

practices, activities, techniques and processes for which the initial objectives of knowledge and experience capture were defined.

Questions or sentences related to levels 1 (Knowledge) and 2 (Comprehension) of Bloom's taxonomy will refer to knowledge that the team members of the project possess or should possess, and that should be applied at the moment of carrying out their project activities. Questions of levels 3 (Application) and 4 (Analysis) refer to the application of previous knowledge or experience that team members already have and that will be applied as they carry out their project activities. Questions of levels 5 (Synthesis) and 6 (Evaluation) relate to the synthesis and evaluation of experiences that project team members acquire when performing their project activities.

For elaborating the reflective questions, the following elements are considered:

- The concepts related to the area of knowledge, activity, technique or software process respect of which it is intended to motivate reflection.
- The knowledge and experience capture objectives that were defined in the Initiation phase.
- The different levels of Bloom's taxonomy described above, with their corresponding keywords that reflect the cognitive operations associated with each level. Lists of keywords can be found in [Bloom, 56] and [Cecil, 95].

For instance, consider the purpose to capture experiences related to the "interview" technique for requirements elicitation, with the goal of improving the interaction among software engineers and stakeholders. Table 1 presents examples of reflective questions and sentences for each level of Bloom´s Taxonomy for some concepts associated to this technique.

Knowledge area/subarea	Requirements engineering / Requirements elicitation
Technique	Interview
Associated concepts: Choosing the interviewee, planning the interview, knowledge of interviewee's terminology, types of questions to be asked, functional and non functional requirements.	
Questions or sentences	
Level 1: Knowledge – Cognitive operation: Listing *Make a list of the steps to follow in order to **plan the interview**.*	
Level 2: Comprehension – Cognitive operation: Comparing ***Compare** the different types of questions that you can make to ask the interviewee.*	
Level 3: Application – Cognitive operation: Selecting *What criteria must be taken into account to appropriately **select** the people to be interviewed?*	
Level 4: Analysis – Cognitive operation: Assessing ***Assessing** the result of the interview, what difficulties and unexpected things do you encounter in your interaction with the interviewee?*	
Level 5: Synthesis – Cognitive operation: Modifying *What aspects of **planning the interview** do you think should be **modified** for a future instance?*	
Level 6: Assessment – Cognitive Operation: Judging *How do you **judge** the process followed in the interview with regard to the identification of **functional** and **non functional requirements?***	

Table 1: Examples of reflective questions or statements

4.2 The Experience Elicitation Workshop

This workshop, which takes place in the Elicitation phase, is a collective reflection and analysis of the knowledge and experiences initially captured in the reflective guides. The purposes of this workshop are:

- To give team members an opportunity to know the answers given by the other team members in order to identify similarities and differences in opinions, viewpoints and lived experiences in accomplishing their project tasks.
- To discuss and reach an agreement on those opinions and viewpoints in order to derive lessons learned and proposals of best practices.

In the workshop the participants are team members who carried out the project tasks related to the software practices and processes preselected in the Initiation phase.

One of the inputs for the accomplishment of the workshop are the answers given by the team members to the questions included in their reflective guides. With these answers, a summary document is elaborated which will include, for each question or sentence, the different reflections registered by the respondents. Another important input for the accomplishment of the workshop is the experience and tacit knowledge of the participants, which, ideally, should become explicit during the workshop.

The workshop begins by communicating to participants the objectives of the workshop, the topics to be discussed, and the general procedure to be followed. Afterwards, the summary document mentioned above will be distributed to the participants in order to know the answers and reflections made by others.

When this activity concludes, an analysis and collective reflection follows, by the order in which reflective questions appear in the reflective guides. It is in this stage where participants report and share their experiences on the projects activities made, contribute with new elements and details that were not explicitly included in the answers to the reflective questions, and give their points of view, reflections and commentaries.

The workshop´s facilitator will take part in this activity, registering the contributions made by the participants, orienting the discussion and enabling the participation of all of them. The workshop concludes with the draft of a summary of contributions made by the participants, from which lessons learned and proposals of best practices will be elaborated in order to incorporate them to the repository of lessons learned and best practices.

This workshop should take place after the reflective guides are collected and processed as explained above. It is worth noting that even though a team member cannot participate in it, his/her experience has already been captured in his/her reflective guide and can be taken in consideration in the collective analysis that take place during the workshop.

4.3 The Repository of Lessons Learned and Best Practices

The Repository of Lessons Learned and Best Practices is constructed from the knowledge and experience acquired by project team members during the accomplishment of their project activities. That knowledge and experience was

elicited and captured during the knowledge and experience elicitation workshop, as explained in the previous paragraph.

This repository has a tree-like structure, following the Software Engineering Body of Knowledge key knowledge areas and sub-areas [Abran, 04]. With this structure, shown in part in figure 3, each leaf of the tree points to lessons learned and/or best practices related to its corresponding knowledge area. We choose the SWEBOK because, at this point, it provides a consensually validated characterization of the bounds of the software engineering discipline.

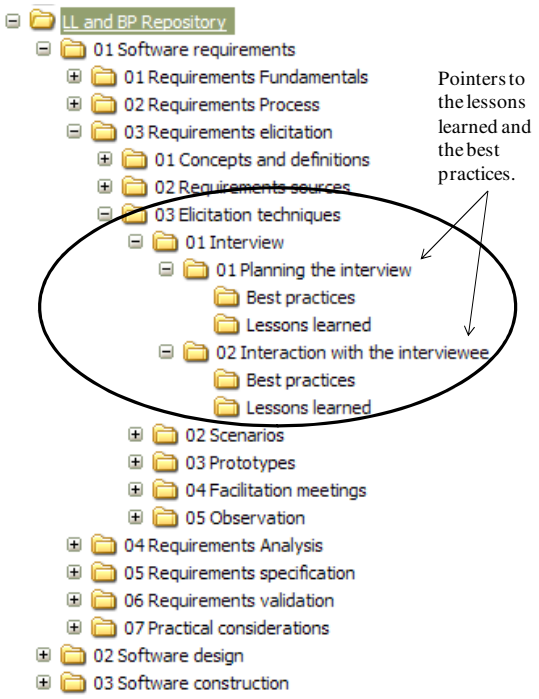

Figure 3: Structure of the LL and BP repository

With this organization, any team member in a new project will be able to navigate the repository and find the lessons learned and best practices that are relevant to the project activities he or she will be responsible to perform. However, this way of disseminating the knowledge existing in the repository strongly depends on the interest and the goodwill of team members for searching those knowledge and experiences and, thus, there is no way to ensure that the lessons learned and the best practices will be used in new projects. To overcome this situation, when the reflective guides are assigned to team members they will include references to the lessons learned and best practices in the repository (if they already exist) related to the project activities they are going to perform. In this way, we enforce team members to be aware and, more important, to apply, the knowledge and experiences gathered in previous cycles. Going back to figure 1, this enforcement is represented by the arrow that goes from the Repository to the “Software project activities”.

Going through several iterations over the same project activities, those lessons learned and best practices can be continuously refined and, when they became stable, the knowledge and experiences they represent can be formally integrated into the organizational software process specification. For doing this, in future iterations of the model, specific questions or sentences can be included in the reflection guides aimed to gather the opinion of team members in relation to, for instance, how useful a lesson learned had proven to be in at particular task or situation, or how good a proposal for best practices really is. Going back again to figure 1, the arrow that goes from the Repository to "Software practices and process in use" represent the way in which knowledge and experience captured in previous projects forms the base for improving software practices and processes in use in the organization.

The simplest way to build the repository is to organize it like a hierarchy of folders in a file system (as shown in Figure 3) and to have each lesson learned or best practice as a separate document, filed in the corresponding folder. More sophisticated ways to create and manage that repository can be based on specialized software, being it a commercial package or a custom-builded one. As we explain at the end of the article, this second alternative is under development at the Software Engineering Department of Universidad ORT Uruguay.

4.4 The Knowledge and Experience Management Group

As the proposed model is intended to give an organization a way to manage its knowledge and experience in software engineering, it is expected that the organization establish a dedicated team for that purpose. We propose this team to have permanent members and ad-hoc or temporary members.

The permanent members are the ones that will take care of implementing the phases and tasks of the mode by, for example, preparing and distributing the Reflective Guides, organizing and conducting the Experience Elicitation Workshop, elaborating the lessons learned and best practices documentation, and populating and updating the Repository.

On the other hand, the temporary members should be experts in their respective software engineering fields and have god knowledge of the software practices and process in use in the organization, and of the areas in which these practices and process require improvements. These temporary members should participate in defining the knowledge and experience objectives, defining the types of question or sentences to include in the reflective guides, and in identifying the lessons learned and proposals of best practices that come out from the Experience Elicitation workshop.

5 Integration of the model into software projects and process improvement activities

The phases and tasks of the proposed model integrates both to software project activities and to the activities of improving the practices and software processes in use, so that the knowledge, learning and experience gained by project team members during the implementation of project activities can be used as a basis for the development and incorporation of improvements to the practices and processes

software in use in the organization. The moment of integration will depend on the phase of the model in which we are in each instance, as graphically shown in figure 4:

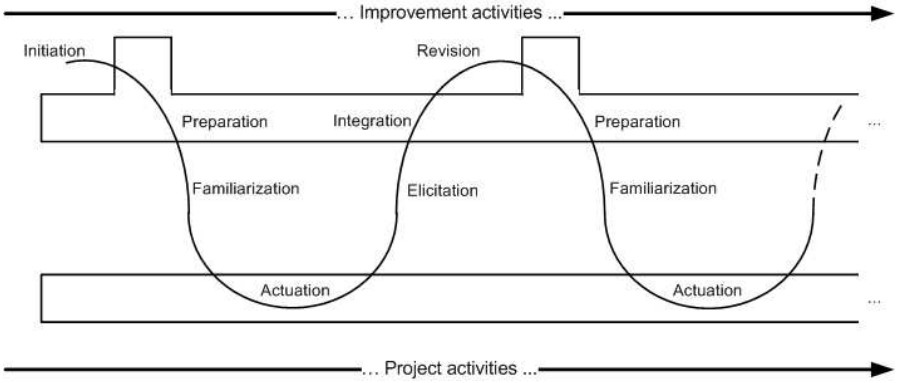

Figure 4: Integration to project and to process improvement activities

5.1 Integration to project activities

The integration of the model and the software project activities occurs fundamentally during the Actuation phase. In effect, the reflective guides contain reflective questions and sentences relative to the projects activities to execute, techniques to apply or processes to be followed by the members of project teams and the intention of these questions or sentences is to motivate and to orientate the analysis and the reflection on the practical execution of those activities, techniques and processes.

5.2 Integration to process improvement activities

The integration of the model to the software practices and process improvement activities essentially occurs in the phases of Initiation, Preparation and Integration. In the Initiation phase is where, from the improvement objectives established by the organization, the software practices and processes to improve are selected, and where the objectives of learning and knowledge creation are defined. In the phase of Preparation is where these objectives of learning and of creation of knowledge are translated in the questions or sentences of reflection to include in the reflection guides. These questions or sentences should aim precisely to motivate the analysis and reflection on those aspects of the software practices and processes that are being tried to improve. Finally, in the Integration phase is where the new knowledge and experience acquired is incorporated to the Repository of lessons learned and best practices and is used as a base for the reformulation of the software practices and processes used in the organization.

6 Implementation of the Model

In order to study the feasibility and usefulness of our model, we implemented it at ORT Software Factory (hereafter, ORTsf), an academic unit within the Software Engineering department of the University ORT Uruguay.

At ORTsf, undergraduate students in the last semester of their Systems Engineer career participate in software projects that have actual customers, that is, organizations external to the University that acquire the software products developed at ORTsf. For every project, team members have to perform all the usual software engineering processes, from requirements elicitation to the final delivery of the product. For this, the roles of requirements engineer, architect, developer, and project manager, among others, are performed by the members of each project team, according to their preferences or based on their previous working experience.

The strategy chosen was a case study. A case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident [Yin, 03]. These characteristics contrast, for example, with the ones of experiments, which are conducted when the researcher wants control over the situation with direct, precise, and systematic manipulation of the behavior of the phenomenon to be studied [Wohlin, 00]. In this sense, and according to Sjøberg, Dyba and Jørgensen [Sjøberg, 07], while an experiment deliberately divorces a phenomenon from its context, the case study aims deliberately at covering the contextual conditions.

For this study, the conditions of the context are particularly important to be taken into account due to the fact that we aim to study the proposed approach and the reflective guides as elements embedded into the daily working activities of the members of a real software team project working in real software development. It is important to consider the fact that a statistical study was not applicable to our case because our focus is mainly qualitative, and the population of available projects at the moment of doing the study was limited to only four projects that matched the desired characteristics of being actual software development projects and with real customers.

6.1 Implementation process

As stated above, four independent software development projects were considered in this study, carried out by 15 students (see Table 2). The project teams were integrated by 3-5 students and had a professional support member who acted as a tutor in each project. Of the fifteen student members of the project team, twelve of them have real experience in developing software or in activities related to the IT. This is a general characteristic among the University students in Uruguay, who also work in the software industry at the same time as they do their higher education careers.

Name	Description	Team size
COODESOR	Management system for a dentistry medical organization	4
GESA	Management system for the Uruguayan Accreditation organism	3
InvPortal	Web site for the Office of Development of the Private Sector (Financial and Economic Ministry of Uruguay)	3
SCPI	Investment projects follow up and control System (Financial and Economic Ministry of Uruguay)	5

Table 2: Projects selected for the empirical study

The working condition for the teams was to work together on-site (in the facilities of the University) for at least 10 hours weekly, in order to promote team cohesion and in order to have a similar working ambience to that of a software organization. The remaining 30 expected weekly hours in the project, the students had the freedom to work in the University or in any other alternative place at their choice. It is important to highlight that the selected projects were not conceived with the specific purpose of research in itself, because these projects had their own agenda and their own deadlines and objectives, which were agreed beforehand with their respective customers.

Based on historical data available at ORTsf about the software practices that are usually evaluated as "deficient" when performed by the project teams, in the Initiation phase we decided to apply the model "ele" with the intention of obtaining lessons learned and proposals of best practices for improving the following software requirement engineering activities: (i) planning interviews with stakeholders and (ii) interaction with stakeholders during the interviews.

Even though we selected these two requirements engineering activities, it is worth to mention that the model does not impose any restriction regarding the project tasks or the software engineering process to be applied to.

For the next phase (Preparation), and for the associated software requirements activities, the corresponding reflective guides were elaborated, containing nine questions. The reflective questions and statements included in the guides were elaborated by the first author in cooperation with a member of the ORTsf staff who, based on working experiences with other former team projects, has good knowledge about the specific aspects of the chosen practices that are important to take into account. Since the goal is to capture experience, and not to define how to plan and handle interviews, the reflective questions and statements are independent of the particular application domains of each software product developed. Another consideration taken into account at the time of defining the reflective questions was the actual knowledge, skills and experience that team members posses about the selected practices. As the participants are undergraduate students in the last semester of their careers, the questions and statements were tailored according to the knowledge and skills they acquired during their regular courses on software requirements engineering.

Once elaborated, the guides were given to the requirements engineer of each of the four projects selected for the study, prior to start the requirements engineering process. This was done during a brief meeting in which we explained them the purpose and content of the guides, and how they are supposed to use those guides as part of their requirements engineering activities.

In the next phase, Actuation, the requirements engineer of each of the projects used the reflective guide while preparing and carrying out the assigned tasks and, during that time, they answered the corresponding reflective questions.

Once the requirements engineering activities were finished and the members of the project team answered all the questions, the guides were collected back for their analysis. Due to their extension, extracts of the answers given for questions 1 and 9 are included below. The complete reflective guides, with all the answers, can be obtained from the first author.

Question 1 corresponds to level 4 (Analysis) of Bloom's taxonomy. Excerpts from the answers are presented in Table 3:

1. What aspects do you consider as important to take into account when planning an interview?	
COODESOR	*I believe that the most important thing in our case was the specific coordination of the interview. It was very good to generate a list of possible questions or topics we had to approach, as a kind of checklist of those aspects that we had to focus on. To inform the interviewee about the topics and the content of the interview in order that the interviewee could reach a clear outline of the topics to be discussed.*
GESA	*To make a list of questions or a guide for the interview; this will be different depending on the person we are going to interview.*
InvPortal	*To make a list of questions or a guide for the interview; this will be different depending on the person that we are going to interview. Before selecting the person who we will interview, it would be useful to speak to a Superior in order to be sure that we are interviewing the right person. Another important point is that the record of the meeting should be done as promptly as possible, as time passes and we can lose "small" details that ultimately are big requirements for the system. Also it is useful to send the client the questions or the topics that are going to be discussed in the meeting, in order to prepare it.*
SCPI	*It is necessary to plan the scope of the interview and the topics that are going to be treated. Make a list with the most important questions, without having to follow step by step, but to make sure ourselves to clarify all the points for which we request the interview.*

Table 3: Answers to question 1

Question 9 corresponds to level 6 (evaluation) of Bloom's taxonomy. Excerpts from the answers are presented in Table 4.

9: Among the activities related to requirements engineering that took place in this project, which ones do you think should be improved next time?	
COODESOR	*A possible improvement is to start with an initial instance of knowledge of the client and other stakeholders before carrying out the interview.*
GESA	*What I should improve for next interviews is time management, because the majority of the times, just for being polite to the customer, I allowed him/her to spread in irrelevant things for the project. This made the meeting longer than planned, without adding anything productive to the interview. I suppose this is due to the lack of experience in interviews.*
InvPortal	*The activity that caused most of the problems was the documentation of everything we extracted. Provided that many tools available in the institutional web pages to take forward the requirement engineering, at first we decided to use the majority of them. But by means of chats with the tutor of role, he advised us that to support all the documents would take time and that is necessary to consider a costs/benefits relation and to be able to realize documents that contribute to the client and to us.*
SCPI	*The only problem was the scarce availability of time by our customer in order to concrete an interview.*

Table 4: Answers to question 9

Once the requirement engineering tasks were done, and with the answers to the reflective questions as an input, we carried out the Experience Elicitation Workshop as the central activity in the Elicitation phase. During the workshop, the requirements engineer of each project discussed the answers given and made new contributions to the knowledge and experiences that were not previously included in the answers to the reflective guides.

The conclusions of the workshop for the two topics defined at the Initiation phase (*interaction with the interviewees* and *planning interviews*) are shown in tables 5 and 6, respectively.

Topic 1: Interaction with the interviewee	
2.1	During the interview, the interviewers must direct their attention to different aspects. For instance, to the topics that are approached, the dynamics of the interview, aspects of the business and possible needs of the user. It is very important that the interviewee does not possess any objection for recording the interview so that the attention centers on the key aspects and does not turn aside towards recording aspects.
2.2	If more than one interviewer will take part in the interview, they should participate together, as a group, to enable an organized and uniform way of conducting the interview.
2.3	Familiarize yourself with the language (“jargon”) of the interviewee.

Table 5: Workshop results for "Interaction with the interviewee"

Topic 2: Planning the interviews	
1.1	Before carrying out the interview it is essential to gather information about the business of the client. To do some research about the client and its domain. This will allow the interviewer to go to the interview with knowledge that will facilitate the comprehension of the problems and needs of the customer.
1.2	Make a list of the questions to be asked to the interviewee and send it before the interview. This will warn the interviewee about the topics to be discussed during the interview, in order the interviewee to be prepared properly and to assure that all the information and supporting documentation is available.
1.3	Elaborate a script of the interview based on the number of respondents and the proposed duration of the interview. This will allow establishing an appropriate order for conducting the interview and, in addition, will help to discuss all the topics, without missing one.
1.4	Establish the duration of the interview and inform it in advance to the interviewee. Set the start time and the finish time and try always to respect these hours. If the end of the interview comes without having discussed all the topics under consideration, try to schedule a new one quickly to conclude the missing topics.
1.5	Write down the results of the interview immediately after its conclusion in order not to forget details or aspects worked during the interview.

Table 6: Workshop results for "Planning the interview"

From these conclusions, lessons learned and proposals of best practices were elaborated that, in the upcoming phase of Integration, were added to the Repository of lessons learned and best practices, whose structure was presented in sub-section 4.3.

6.2. Process Timing

As experience indicates, team members usually don't like their activities to be interrupted by "additional" tasks, especially for knowledge management. Because of this, we also wanted to know how much time the participants spent in answering the questions in the reflective guides as a way to measure the impact this "additional" activity has in their project-specific activities and in the total time of the projects.

In table 7 we present the times (in hours) devoted by the requirement engineer of each project in answering the questions in the reflective guides.

Project	Timing (in hours)
COODESOR	1.33
GESA	1.27
SCPI	0.77
InvPortal	1.17

Table 5: Times devoted to answer the reflective guides

To know the impact these additional activities had in the project activities, we proceeded to calculate the percentage of time spent in the requirements engineering activities and the total amount of time involved in the project. From the revision of the documents of the projects of the competing groups the following timing used in the overall project and the engineering requirements activities were obtained; in table 8 the timing is expressed in hours.

Project	RE activities (in hours)	Project duration (in hours)
COODESOR	372	1371
GESA	233	1262
SCPI	262	2912
InvPortal	27	1480

Table 6: Total times of project activities

Table 9 shows the percentages of time dedicated to answering the reflective guides in relation to the time consumed by the activities of requirements engineering and total project time.

Project	% related to RE activities	% related to project duration
COODESOR	0.36 %	0.09 %
GESA	0.55 %	0.10 %
SCPI	0.29 %	0.03 %
InvPortal	4.33 %	0.08 %

Table 7: Incidence of answering the reflective guides in relation to projects times

Based on the data presented in the previous three tables, the impact of answering the questions in the reflective guides is less than 1% of the time devoted to the whole software requirements process in each project with the exception of project InvPortal. The reason behind this behaviour is that the requirements engineering process for project InvPortal took only 27 hours, but answering the reflective guides took about the same time as the other three projects (as shown in Table 7). However, if we take into consideration the total duration of each project (shown in Table 8), the impact of answering the reflective guides was less than 0.1% for all of them.

The costs in time of recording the answers, shown in table 7, are not the only ones to take into account for evaluating the time consumed by the entire approach. Other costs include the time required for preparing the reflective guides, for analyzing the answers after the guides are returned back, for preparing and carrying out the workshop, and for the elaboration of the lessons learned and proposals of best practices, beside the time spent in the brief meeting explained above. All of these times, in hours, are presented in table 10.

Other activities	Times consumed (in hours)
Reflective guides elaboration	1.5
Brief meeting	1.0
Analysis of the answers	6.0
Workshop process	2.5
LL and BP elaboration	1.5
Total	**12.5**

Table 8: Times devoted to the other activities

As we have data of only these four projects, we are unable to do a statistical treatment of these data in order to derive more general conclusions from them. We take these data only as a primary estimation of the effort that implies the use of the reflective guides as part of project activities.

7 Conclusions and future works

In this paper we presented the model "ele" for managing the knowledge and experience that team members acquire during software projects, with its phases and tasks integrated into daily project activities. The purpose of the model is to guide knowledge and experience management activities in order to capture and manage lessons learned and proposals of best practices that will enable an organization to preserve them for future use.

The proposed model provides a complete life cycle for knowledge and experience management, from the initial definition of objectives for the experience to be captured, to the final elaboration and dissemination of lessons learned and best practices.

A central element in the model is what we called "reflective guides". By integrating them to the project tasks to be carried out, the process of experience capture occurs earlier than with existing methods such as project postmortem analysis ([Birk, 02]), post-project revisions ([Harrison, 03]), legacy sessions ([Cooper, 07]), and experience postmortem reports and reviews ([Dingsoyr, 01]) mentioned in section 1, avoiding the risk of these experiences being lost if, after project completion, the team members abandon the organization or are assigned to new projects, taking with then the acquired experience.

Our model goes beyond the mere capture of experience, providing specific directions on how to use the captured experience for identifying lessons learned and proposals of best practices that are later integrated to a general repository to preserve them for future use. The structure given to this repository makes it easy for team member to locate existing lessons learned and best practices, enabling this way the dissemination of knowledge and experience to the rest of the organization and its reuse in new software projects.

By implementing the model, an organization can achieve an organized and systematic way of managing its software engineering knowledge and experiences, covering the usual knowledge management activities generally found in the literature, such as identification, capture, preservation, dissemination and reutilization.

An implementation of the model in a software organization was also presented. The study carried out shows that the use of the reflective guides enabled the capture of experience gathered by team members during the execution of project tasks. This contrasts with the way this experience is captured by applying traditional methods such as project postmortem analysis, as commented in the introduction.

The subsequent experience elicitation workshop enables requirements engineers to know the experiences lived by others and to make new contributions of knowledge and experiences not initially captured in the guides. As a result of this workshop, lessons learned and proposals of best practices are elaborated and incorporated into the repository of lessons learned and best practices.

As an additional result of exploratory character, the time consumed by the members of the project teams was obtained. Even though we do not have data to compare this timing with the timing necessary to capture experience by using the methods presented in section 2, we can say that the use of reflective guides in particular, and the whole process in general, cannot be considered an overload of work for the member of the team projects. With regard to this issue, we consider necessary to have not only quantitative data about this timing, but to have qualitative data as well, in order to bring into consideration the quality and richness of the results obtained with our approach, compared to other methods. For the post-mortem analysis method in particular, it could be interesting to investigate how it could be enhanced if the answers to the reflective guides were used as an input. By using reflective guides, the questions or sentences included in them can be of help to define the goals of the post-mortem during its Preparation phase, and the Data Collection phase can be considered almost done with the answers given by team members. The only remaining phase would be the Analysis phase, that is, the feedback session conducted to analyze the causes of positive and negative experiences. Additionally, even though some team members cannot finally participate in this later phase, their experiences can be taken into account by considering their answers to the reflective questions.

The implementation of the model also served to define in a precise way the purposes and objectives of each phase, as well as to adjust the procedures for elaborating the different knowledge management artifacts within the model. At the moment of writing this article, the first author is working with two Uruguayan software companies in defining a plan for implementing the model in their environments. We are carrying out an assessment of the software practices and processes in use in both organizations as a previous step to define the objectives for knowledge and experience capture. Implementing the model in these two organizations will give us the opportunity to test it with bigger project teams than the ones considered in the case study.

Another topic that will deserve further research is how to tailor the model for applying it in a distributed software development environment, particularly for the step of the Experience Elicitation Workshop. At first sight we are considering the use of videoconferencing technology for this purpose which also will give us useful information to compare it with a face-to-face meeting.

On the technical and practical side, a software tool is being developed in ORTsf to support the different phases and tasks of the model, which will also interface with the repository of lessons learned and of best practices. Salient features of this tool will be to allow team members to read and add comments to the reflections and answers given by the other, and also to enable users of the repository to include comments regarding the lessons learned and the best practices and to rate them in order to create a ranking of the most useful or interesting ones as a way to give a recognition to the respective authors.

References

[Abran, 04] Abran, A., Moore, J., (eds.) Guide to the software engineering body of knowledge, IEEE Computer Society, Los Alamitos, California, 2004

[Alagarsamy, 07] Alagarsamy, K., Justus, S., Iyakutti, K. The knowledge based software process improvement program. A rational analysis, Proceedings of the International Conference on Software Engineering Advances (ICSEA 2007), 2007 pp. 61.

[Antunes, 07] Antunes, B., Seco, N., Gomes, P., Knowledge management using semantic web technologies: An application in software development, Proceedings of the Fourth International Conference on Knowledge Capture, 2007 187-188

[Aurum, 08] Aurum, A., Daneshgar, F., Ward, J. Investigating Knowledge Management practices in software development organizations. An Australian experience, Information and Software Technology 50, 2008, pp. 511–533.

[Azuma, 03] Azuma, M., Coallier, F., Garbajosa, J., How to apply the Bloom taxonomy to software engineering, Proceedings of the Eleventh Annual International Workshop on Software Technology and Engineering Practice (STEP '03), 2003, pp. 117-122

[Babar, 09] Babar, M., Dingsøyr, T., Lago, P., van Vliet, H. Software architecture knowledge management, Berlin, Springer, 2009

[Basili, 94] Basili, V., Caldeira, G., Rombach, H. The experience factory, in J. Marciniak (Ed.) Encyclopedia of software engineering, New York, J. Wiley & Sons, 1994, pp. 469-476

[Basili, 01] Basili, V., Lindvall, M., Costa, P. Implementing the Experience Factory as a set of experience bases, Proceedings of the 13th International Conference on Software Engineering and Knowledge Engineering, 2001, pp. 102-109

[Birk, 02] Birk, A., Dingsoyr, T., Stalhane, T., Postmortem: never leave a project without it, IEEE Software, 19 (3), (2002) pp. 43-45

[Bjørnson, 07] Bjørnson, F. Knowledge management in software process improvement, Doctoral thesis, Department of Computer and Information Science, Norwegian University of Science and Technology, 2007

[Bloom, 56] Bloom, B., Engelhart, M., Furst, E., Hill, W., Krathwohl, D., Taxonomy of educational objectives. Handbook I: Cognitive domain, David McKay, New York, 1956.

[Bourque, 03] Bourque, P., Buglione, L., Abran, A., April, A., Bloom's taxonomy levels for three software engineering profiles, Proceedings of the Eleventh Annual International Workshop on Software Technology and Engineering Practice (STEP '03), 2003, pp. 123-129.

[Briand, 03] Briand, L., On the many ways software engineering can benefit from knowledge engineering, Proceedings of the 14th International Conference on Software Engineering and Knowledge Engineering, 2003, pp. 3-6

[Burge, 02] Burge, J., Brown, D. Integrating design rationale with a process model, Proceedings of the Workshop on Design Process Modelling, Artificial Intelligence in Design '02, Cambridge, UK., 2002

[Cecil, 95] Cecil, N., The art of inquiry, Winnipeg, Peguis, 1995

[Chau, 05] Chau, T., Maurer, F. A case study of wiki-based experience repository at a medium-sized software company, University of Calgary, Department of Computer Science, 2005

[Cooper, 07] Cooper, L., Converting project team experience to organizational learning, Proceedings of the 40th Hawaii International Conference on System Sciences, 2007, pp. 195

[Desouza, 05] Desouza, K., Dingsoyr, T., Awazu, Y., Experiences with conducting project postmortems. Proceedings of the 38th Hawaii International Conference on System Sciences, 2005, pp. 233c.

[Dingsoyr, 01] Dingsoyr, T., Moe, N., Nytro, O., Augmenting experience reports with lightweight postmortem reviews, in F. Bomarius, S. Komi.Sirvio (Eds.): PROFES 2001, LNCS 2188, 2001, pp. 167-181.

[Edwards, 03] Edwards, L., Coaching: the latest buzzword or a truly effective management tool?, Industrial and Commercial Training, 35 (7), 2002, pp. 298-300

[Gupta, 04] Gupta, J., Sharma, S., Creating knowledge-based organizations, Harshey, Idea Group Inc., 2004

[Harrison, 03] Harrison, W., A software engineering lessons learned repository, Proceedings of the 27th Annual NASA Goddard/IEEE Software Engineering Workshop, 2003, pp. 139.

[IEEE, 90] IEEE Computer Society, IEEE standard glossary for software engineering terminology, IEEE, Piscataway, 1990

[Jedlitschka, 01] Jedlitschka, A., Althoff, K., Decker, B., Harrtkopf, S., Nick, M. Corporate information network (COIN): The Fraunhofer IESE experience factory, 2001, pp. 54-60

[Johansson, 99] Johansson, C., Hall, P., Coquard, M. Talk to Paula and Peter; they are experienced. The experience engine in a nutshell, Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering, Learning Software Organizations, Methodology and Applications, 1999, pp. 171-185

[Keegan, 01] Keegan, A., Turner, J., Quantity versus quality in project-based learning practices, Management Learning 32(1), 2001, pp. 77-98

[Komi-Sirvio, 02] Komi-Sirvio, S., Mantyniemi, A., Seppanen, V., Toward a practical solution for capturing knowledge for software projects, IEEE Software, 19 (3), 2002, pp. 60-62

[Koskinen, 01] Koskinen, K. Tacit knowledge as a promoter of success in technology firms, Proceedings of the 34th Hawaii International Conference on System Sciences, 2001

[Kruchten, 05] Kruchten, P., Lago, P., Van Vliet, H., Wolf, T. Building up and exploiting architectural knowledge, Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture, WICSA 2005. IEEE Computer Society, 2005, pp. 291–292

[Kukko, 08] Kukko, M., Helander, N., Virtanene, P. Knowledge management in renewing software development process, Proceedings of the 41th Hawaii International Conference on System Sciences, 2008, pp. 332

[Maier, 08] Maier R., Thalmann S., Bayer F., Krueger M., Nitz H., Sandow A., Optimizing assignment of knowledge workers to office space using knowledge management criteria, Journal of Universal Computer Science, 14 (4), 2008, pp. 508-525

[Pettersson, 09] Pettersson, U., Success and Failure Factors for KM: The Utilization of Knowledge in the Swedish Armed Forces, Journal of Universal Computer Science, 15(8) 2009, pp. 1735-1743

[Rus, 02] Rus, I., Lindvall, M. Knowledge management in software engineering, IEEE Software, 19 (3), (2002) 26-38

[Schneider, 02] Schneider, K., von Hunnius, J., Basili, V. Experience in implementing a learning software organization, IEEE Software, 19 (3), 2002, 46-49

[Scott, 03] Scott, L., Stalhane, T., Experience repositories and postmortem, Workshop Learning Software Organizations, 2003

[Sjøberg, 07] Sjøberg, D., Dyba, T., Jørgensen, M., The future of empirical methods in software engineering research, In 2007 Future of Software Engineering, International Conference on Software Engineering. IEEE Computer Society, Washington, DC, pp. 358-378

[Tang, 06] Tang, A., Babar, M., Gorton, I., Han, J., A survey of architecture design rationale Journal of Systems and Software, 79(12), 2006, pp. 1792–1804

[Wohlin, 00] Wohlin C., Runeson P., Höst P. Experimentation in software engineering: An introduction, Kluwer, Boston, 2000

[Wohlin, 03] Wohlin, C. Applications of knowledge management in software engineering, in: R. Jeffery, C. Wohlin, M. Handzic (Eds.), Managing software engineering knowledge, Berlin, Springer, 2003, pp. 177-180

[Ye, 06] Ye, Y. Supporting software development as knowledge intensive and collaborative activity, Proceedings of (WISER'06), 2006, pp. 15-22

[Yin, 03] Yin, R. Case study research. Design and methods, Sage, Thousand Oaks, 2003

[Zedtwitz, 02] Zedtwitz, M., Organizational learning through post-project reviews in R&D, R&D Management, 32(3), 2002, pp. 255-268

[Zhu, 07] Zhu, L., Staples, M., Gorton, I. An infrastructure for indexing and organizing best practices, Proceedings of the 2nd International Workshop on Realizing Evidence-based Software Engineering (REBSE '07), 2007, pp. 4