Computational Intelligence, Volume 17, Number 2, 2001

SYSTEMATIC MAINTENANCE OF CORPORATE EXPERIENCE REPOSITORIES

MARKUS NICK, KLAUS-DIETER ALTHOFF, AND CARSTEN TAUTZ

Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany

Experience-based continuous learning is essential for improving products, processes, and technologies in emerging as well as established areas of business and engineering science. This can be facilitated by case-based organizational learning through capturing relevant experience in the form of cases for reuse in a corporate experience repository. For obvious reasons, learning from experience needs to be a permanent endeavor. Thus an organization has to handle a "continuous stream of experience." Consequently, such an experience repository requires maintenance. This should not simply happen ad hoc but in a systematic manner. To make competent decisions about maintenance, the experience base and its usage have to be analyzed (i.e., evaluated). To improve maintenance itself, it is necessary to learn about it. For this purpose, we identify the relevant tasks for maintaining an experience repository and the responsibilities of the roles involved. Maintenance addresses not only the actual experience in the form of cases but also the conceptual model and the methods, techniques, and tools that are used for filling and updating the experience repository. To support the roles involved in the maintenance tasks, we provide a flexible, practical maintenance and evaluation framework. This framework provides guidance for the roles. The framework can be combined with other approaches from artificial intelligence, knowledge engineering, and software engineering at different levels of detail. For the practical application of the framework, we describe an integrated technical solution for a corporate experience repository that is maintained using our framework. Furthermore, we discuss the empirical validation of the framework and its implementation.

Key words: experience management, experience factory, experience base, corporate memory, maintenance, evaluation.

1. INTRODUCTION

In all emerging areas of business and engineering science, there is normally a lack of explicit knowledge about the underlying processes, products, and technologies. Usually such knowledge is built up through individual learning from the experience of the people involved. The field of organizational learning tries to increase the effectiveness of individual human learning for a whole organization. Besides improving internal communication (group learning), organizational learning also includes documenting relevant knowledge and storing it (for reuse) in an organizational/corporate memory (van Heijst et al. 1996).

An approach known from software engineering called *experience factory* (EF) (Basili et al. 1994) goes one step further. Knowledge (in the form of processes, products, and technologies) is enriched by explicit experience cases[1] (e.g., explicitly documented lessons that were learned during the practical application of the knowledge). The EF approach explicitly includes collecting, documenting, and storing such experience as experience cases in an experience base (EB), which is an organizational memory for relevant knowledge and experience. This tries to make human "learning from experience" explicit in order to further support repository-based organizational learning. We call this *experience-repository-based organizational learning.*

Even domains where best practice (in the form of processes, products, and technologies) is well documented can benefit from experience-repository-based organizational

Correspondence should be addressed to Markus Nick at the Fraunhofer Institute for Experimental Software Engineering, Sauerwiesen 6, 67661 Kaiserslautern, Germany; e-mail: {*nick, althoff, tautz*}*@iese.fhg.de.*

[1]Note that in the EF literature "experience cases" are also called "experience packages" (Basili et al. 1994). The term *experience case* emphasizes the case nature of these items with respect to case-based reasoning as the technology of choice.

learning because experts usually adapt the best practice when they apply it and gain (more) experience [“situated cognition” (Menzies 1998)].[2] Thus a description of best practice alone would be “outdated” soon. Experience-repository-based organizational learning aims to avoid this by enriching the best-practice description with experience gained in its application. When reapplying some best practice, the expert is provided with the explicit experience gained by his or her colleagues and himself or herself in former applications.[3]

The concept of experience-repository-based organizational learning is closely related to case-based reasoning (CBR): The way experts use and apply knowledge resembles the CBR cycle (Aamodt and Plaza 1994). Experts match a current problem to libraries of best practice to find a suitable practice and the respective experience that is relevant to the current problem context (“retrieve”); adapt the best practice according to the experience to get a suggested solution (“reuse”); apply the suggested solution, check its performance, and derive a confirmed solution (“revise”); and store the new/updated practice and/or the experience gained in the library (“retain”).

Therefore, CBR technology can support experience-repository-based organizational learning especially in the context-sensitive retrieval of best practice (i.e., explicit knowledge about the underlying processes, products, and technologies) and the capturing of experience cases about best practice.

The core idea of the EF, namely, to improve processes, products, and/or technologies based on a “continuous stream of experience,” has the direct consequence that maintaining an EB[4] is an essential task to keep it attractive and useful to preserve and improve its value for supporting (organizational) learning from experience. An EB includes several kinds of knowledge: For buildup and usage, there are the experience cases as well as their underlying structure (conceptual model); for the actual maintenance of the EB, specific maintenance experience (and its conceptual structure) has to be included as well (Menzies 1998). Additionally, organizational issues have to be considered for the EF around the EB.

How such maintenance experience can be used to do “good” maintenance is still an open issue. Because maintenance is always done with a particular goal in mind, “good” maintenance has to attain this goal. For the purpose of this work, the *goal of maintenance* is to *preserve and/or improve the value of the experience base.*

Maintenance of EBs has partly been discussed in the literature (Birk and Tautz 1998; Tautz 2000; Minor and Hanft 2000). DISER (a methodology for designing and implementing software engineering repositories), for instance, includes some basic methods for dealing with the EF/EB maintenance problems (Althoff et al. 1999b), but they remain on a rather abstract level of description. What is still missing is a systematic maintenance framework for EBs that is detailed enough to be applied in practice. In CBR there has already been some work on maintenance (Smyth and Keane 1995; Heister and Wilke 1998; Leake and Wilson 1998; Smyth and McKenna 1998; Zhu and Yang 1999; Minor 2000; Ferrario and Smyth 2000; Reinartz et al. 2000).

[2]“It is only novices who slavishly re-apply accepted practice” (“Knowledge Maintenance: The State of the Art,” Tim Menzies, tutorial at SEKE’99, slide 20).

[3]This notion of experience can be viewed as informal adaption knowledge (for the adaption of best practice). The expert is provided with this adaption knowledge but has to perform the actual adaption by himself or herself. This is especially feasible if the adaption knowledge cannot be formalized or if the effort for formalizing adaption knowledge is (considered or expected to be) too high.

[4]The authors would like to clarify that within this article, *EB maintenance* denotes *case-based reasoner maintenance* because maintenance of the terminology and similarity measure containers, both in the sense of Richter (1998), is always included. In addition, maintenance of the respective maintenance processes is also considered. Thus this contribution focuses on maintenance of the technical system (case-based reasoner) as well as maintenance of the overall organizational system, of which the technical system is a part.

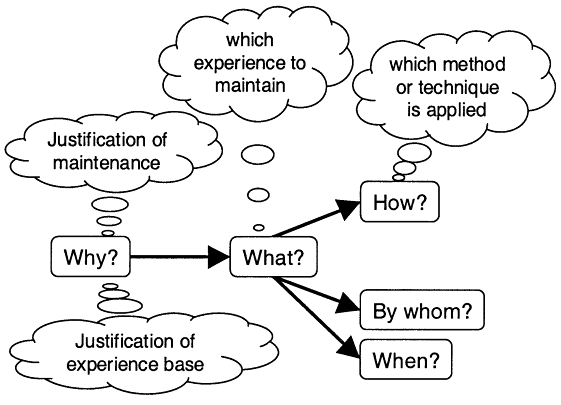
which experience to maintain
which method or technique is applied
Justification of maintenance
How?
Why?
What?
By whom?
Justification of experience base
When?

FIGURE 1. Dimensions of maintaining experience.

A number of factors obviously influence EB maintenance and require flexible strategies for maintenance. For example, different types and numbers of experience cases require different methods and/or tools for recording, updating, and forgetting. Larger EBs and/or a higher numbers of users require more EF staff. High EF personnel turnover requires capturing more of their maintenance experience in the form of methods and tools, etc.

All this shows that an ad hoc approach to EB maintenance is not appropriate. Instead, a systematic approach is required for EB maintenance to ensure "good," well-controlled maintenance. Such an approach must include well-justified decisions (*why*) about the *what* (i.e., which experience to maintain), *how* (i.e., which method and/or tool is applied), *by whom*, and *when* of EB maintenance (Figure 1). The approach must be flexible to deal with the variations of the above-mentioned factors that influence EB maintenance.

In this article we first explain the EF concept and describe the major operation phases "EB buildup" and "regular service" with EB maintenance (Section 2). We identify the relevant tasks for maintaining an experience base (Section 3) and the responsibilities of the roles involved (Section 4). The maintenance addresses not only the actual experience in the form of cases but also the conceptual model and the methods, techniques, and tools that are used for filling and updating the experience base. To support the roles involved in the maintenance tasks, we provide a flexible, practical maintenance and evaluation framework (Section 5), which also provides the glue between the EB's different kinds of knowledge [i.e., the knowledge containers in the sense of Richter (1998)]. The framework is illustrated with a scenario. It can be combined with other approaches from artificial intelligence, knowledge engineering, and software engineering at different levels of detail. For the practical application of the framework, we describe an integrated technical solution for a corporate experience repository that is maintained using our framework and outline its use for an existing EB (Section 6). The empirical validation of the framework and its implementation are discussed in Section 7. Finally, some conclusions are drawn (Section 8).

2. EXPERIENCE FACTORY CONCEPT

The EF concept is rooted in the area of software engineering. In the late 1980s, Basili and Rombach (1988) recognized that software development as an engineering discipline requires the explicit capturing of the state of the practice. To ensure a certain level of quality of software systems (e.g., in terms of reliability), it is a necessity to use standardized processes, i.e., processes that have proven to deliver high quality. However, software development is not a manufacturing discipline where the same product is produced over and over again. Rather, each software development project is different, although similar in many ways to past developments. This means that the standardized processes need to be tailored to project-specific needs. Thus it is not enough to capture the processes. It is also necessary to capture tailoring experience (e.g., in the form of lessons learned).

Once the state of the practice has been documented explicitly, it is also possible to improve it on a continuous basis, e.g., by performing cross-project analyses and then identifying commonalities and differences of the tailored processes. Insights from such analyses may result in additional experience (e.g., rules on when to use which tailoring experience) or the modification of already existing experience (e.g., a change of standard processes). In addition, strategic decisions can be implemented by changing the standard processes. However, cross-project analyses and the implementation of strategic decisions are not the objectives of a software development project. A development project has the goal to deliver some product that fulfills given requirements under given resource constraints (such as time and cost constraints). Therefore, continuous improvement requires a separate, dedicated organizational unit. This unit is called the EF.

There are several principles underlying the EF concept. The most important ones are

- A separate organizational unit (in addition to those organizational units performing "everyday business") is needed for continuous improvement.
- Improvement actions are initiated and performed in accordance with strategic decisions made within the organization (consisting of all organizational units). Thus the improvement actions are systematically derived from the business goals.
- Experience is captured explicitly to be able to analyze and improve it systematically as well as to show improvements to third parties (e.g., customers) using quantitative figures.
- Experience is captured in a systematic, goal-oriented manner. All kinds of experience, which are (potentially) relevant to others, are captured. This includes not only product-oriented experience (such as deliverables and other work products) but also process-oriented experience, which describes *how* the product-oriented experience was/is being developed. Experience irrelevant for others is *not* captured.
- Work supported by the EF is nontrivial in the sense that the same process cannot be applied repeatedly; i.e., the work is knowledge-intensive and cannot be completely automated. Thus part of the work must be performed by humans.

Although originally developed for software engineering, the EF concept is not limited to that domain. It can be used beneficially in any organization dealing with knowledge-intensive tasks where the synthesizing and sharing of experience yields productivity gains. In organizations where it is not possible to talk to everybody on a regular basis (e.g., geographically distributed organizations or organizations with more than 30 employees), the EB can act as a medium over which experience is exchanged (provided everybody has access to it).

2.1. COIN: An Example of an Experience Factory

Founded in January 1996, our institute grew to 120 employees in about 4 years. One characteristic of such fast-growing organizations is the small number of employees who have been in the organization from the beginning. Therefore, this small group of experts becomes a scarce resource as information providers. Hence it is important (1) to provide the less experienced people with default processes and guidelines to jump-start them and (2) to facilitate experience sharing among them to build up their expertise more quickly. Since the size of our institute does not allow everyone to talk to all people on a weekly basis, experience sharing on a personal basis does not work. Therefore, a project named *COIN* was launched.

COIN (corporate information network) is an EB in which all kinds of experience necessary for our daily business (e.g., projects, business processes, document templates, guidelines, observations, improvement suggestions, problems that occurred, and problem fixes that were applied) are stored. Defined processes populate this EB systematically with experience typically needed by our project teams. Dedicated improvement processes analyze problems that have occurred, devise improvement actions to avoid their recurrence, and implement strategic decisions by the institute's leadership. The COIN team (resembling the personnel of the EF) is responsible for performing these processes (Tautz et al. 2000).

The EB is built up incrementally. Besides the listed kinds of experience, which are currently captured, other kinds of experience such as slide presentations, publications, reports, and contracts will be stored in the EB in the future.

The examples show that there are many different kinds of experience to be stored in the EB. In addition, these experience cases are highly interrelated. For example, projects produce deliverables in the form of slide presentations and reports. Slide presentations may be summaries of reports. Observations and problems are gained during a project while a particular business process was performed (context-sensitive experience). Context-sensitive experience is unique in the sense that the same context will not recur. Therefore, people will be searching for experience that has been gained in *similar* contexts. Both the requirement for supporting different kinds of interrelated experience cases and the need for context-sensitive, similarity-based retrieval demand a specialized infrastructure for the EB.

These are common requirements for an EB (Tautz 2000). Our solution to meet these requirements is INTERESTS (intelligent retrieval and storage system) (Althoff et al. 1999a). INTERESTS consists of a general-purpose browser for accessing and presenting the EB contents using a standard Web browser, an EB server synchronizing (and logging) access to the EB, and CBR-Works or orenge from tec:inno, Germany (Schulz 1999), which is used for the actual EB. Each experience case is stored as a CBR-Works/orenge case in the form of (typed) attribute value pairs. Relations between cases are documented using the reference type of CBR-Works/orenge. Figure 2 shows an excerpt of the case base of COIN, which will be used in examples in the remainder of this article. The focus in the excerpt is on lessons learned in the form of guidelines, observations, and problems. The guidelines act as solutions or mitigation strategies for the problems. An observation describes the results of an application of a guideline.

2.2. Life Cycle of an Experience Base

Two major phases of an experience base can be identified: buildup and regular service. During buildup, objectives (including high-level success criteria) and subject areas of the EB need to be identified; the conceptual model (domain model) underlying the experience cases has to be developed; processes for populating, improving, and utilizing the EB have to be defined; and the actual infrastructure must be implemented according to the organization's needs. These parts should be developed using some

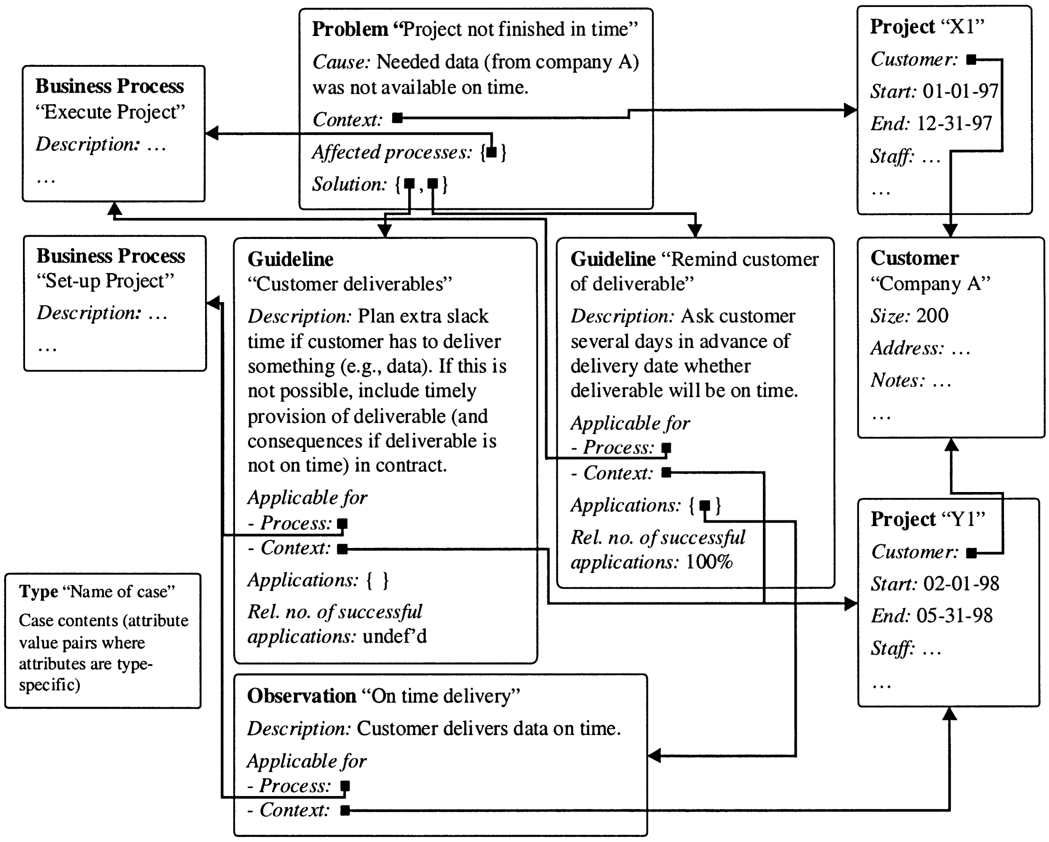
Problem "Project not finished in time"
Cause: Needed data (from company A) was not available on time.
Context:
Affected processes: { }
Solution: { , }
Project "X1"
Customer:
Start: 01-01-97
End: 12-31-97
Staff: ...
Business Process "Execute Project"
Description: ...
Business Process "Set-up Project"
Description: ...
Guideline "Customer deliverables"
Description: Plan extra slack time if customer has to deliver something (e.g., data). If this is not possible, include timely provision of deliverable (and consequences if deliverable is not on time) in contract.
Applicable for
- Process:
- Context:
Applications: { }
Rel. no. of successful applications: undef'd
Guideline "Remind customer of deliverable"
Description: Ask customer several days in advance of delivery date whether deliverable will be on time.
Applicable for
- Process:
- Context:
Applications: { }
Rel. no. of successful applications: 100%
Customer "Company A"
Size: 200
Address: ...
Notes: ...
Project "Y1"
Customer:
Start: 02-01-98
End: 05-31-98
Staff: ...
Type "Name of case"
Case contents (attribute value pairs where attributes are type-specific)
Observation "On time delivery"
Description: Customer delivers data on time.
Applicable for
- Process:
- Context:

FIGURE 2. Excerpt of an experience base with nine cases.

methodology such as DISER [see Tautz and Althoff (2000) and Tautz (2000) for an industrial-strength case study and a detailed description]. Usually, these parts are not developed from scratch. Instead, they are tailored from similar parts used in other organizations. During regular service,

- Actual experience cases are recorded in the EB.
- Success criteria are checked, and—if necessary—corrective actions are taken for improvement toward the success criteria.
- Strategic decisions (e.g., the addition of a new subject area) are implemented by changing some or all of the parts developed during the initial buildup. This activity is comparable with the initial buildup because both start with parts that are tailored to new needs. However, this activity is more complex because already stored experience cases may be affected by the restructuring. In general, stored experience cases may become obsolete, additional experience cases may have to be captured, or existing experience cases may need to be changed contentwise. The latter may require interaction with the original provider or may not be possible at all (e.g., if the experience provider has left the organization or does not remember).

We consider all tasks performed during regular service as *maintenance* of the experience base and thus as maintenance of our CBR system. Therefore, we use a very broad definition of the term that includes—as a subset—techniques and methods for porting a CBR system from one organization to another.

In the subsequent sections we will show that maintenance needs to be planned systematically. To know why and when to do what during the "regular service" phase of an experience base, a maintenance infrastructure and corresponding guidelines and processes need to be developed. This development should be part of the initial buildup. To cover all aspects of maintenance, DISER needs to be extended. For example, the high-level success criteria need to be complemented by low-level measures that indicate deviations from the planned objectives early on when it is still possible to correct the negative trend with a low amount of effort. A maintenance guideline would describe when to collect which measurement data as well as what has to be done in case the data indicate a negative trend. Processes describe how to collect and interpret the data.

3. MAINTENANCE TASKS FOR EXPERIENCE BASES

To make EB maintenance a systematic effort and to facilitate learning about EB maintenance, EB maintenance tasks must be defined. The overview of EB maintenance tasks—presented in this section—aids in determining what activities can be supported by methods, techniques, and tools.

The tasks involved in EB maintenance can be separated into two levels. First, there is a strategic level with a long-term perspective. This level addresses, for example, which subject areas are to be maintained, which business goals are to be attained by the EF, and general guidelines for managing maintenance (e.g., resource allocation) to be considered. Second, there is an operational level with a short- to midterm perspective. This level deals with the actual maintenance operations on the experience base as well as the decision making about these maintenance operations and their organization. For the purpose of this article, we focus on the tasks concerning the operational level because most of the effort required for maintaining an EB is at the operational level. In the following we first give an overview of the tasks and their relationships (as depicted in Figure 3) and then describe the tasks in more detail.

On the operational level, two loops are implemented: the maintenance loop and the feedback loop (Figure 3).

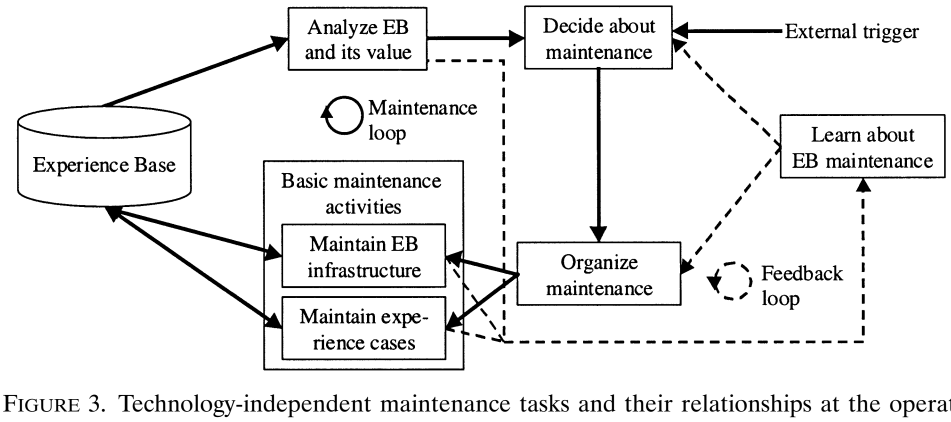

FIGURE 3. Technology-independent maintenance tasks and their relationships at the operational level.

- The actual maintenance is done in the maintenance loop and consists of four steps: (1) analyzing the EB and its usage, (2) deciding about maintenance, (3) organizing maintenance, and (4) maintaining the EB infrastructure and experience cases. The task of analyzing the EB (infrastructure, contents) and its usage delivers results that provide the basis and input for making maintenance decisions about the actual basic maintenance activities (i.e., which experience cases or parts of the EB infrastructure shall be maintained). Based on the decisions about maintenance, the basic maintenance activities have to be organized (i.e., scheduled and assigned to persons).
- To improve future maintenance by basing maintenance decisions not only on evaluation but also on special maintenance experience from former maintenance and evaluation activities, a feedback loop is introduced. In this loop, experience about EB maintenance that is gained during analysis and basic maintenance activities is fed back to the decision-making and organizing tasks.

Although several activities may be performed for each task, all activities of one task aim at the same objective. In the following, the objectives of the tasks are described along with exemplary activities:

- The objective of *analyzing the EB (infrastructure, contents) and its usage* is to find potential for improving the EB. The analysis can result in various insights. For example, query logs (containing both queries issued and resulting experience cases the user looked at) can be analyzed to monitor the coverage of subject areas. If many queries are issued for a given subject area but users look only at very few experience cases, the analysis result will yield "low coverage of the subject area" (Althoff et al. 1999b).
- The objective of *deciding about maintenance* is to determine which basic maintenance activities are to be performed. To do so, each insight from the EB analysis (with improvement potential) is considered. Adequate basic maintenance activities with an effort estimation for their performance are identified. If possible, the expected benefits are also estimated to provide a basis for organizing the maintenance. Such decisions are based on evaluation results or on special maintenance experience ("maintenance guidelines"). Many types of basic maintenance activities are conceivable: updating or reorganizing existing experience, removing superfluous experience, or recording new experience that has been made available through ongoing projects (Nick and Althoff 2000; Tautz 2000; Althoff et al. 1999a). For example, the EB analysis may show that the coverage of a subject area is too low. This leads to the decision that more experience shall be collected for this subject area by performing project analyses for more projects in this subject area.
- The objective of *organizing maintenance* is to determine when which basic maintenance tasks ("maintain experience cases" and "maintain EB infrastructure") are to be performed and by whom. For example, the maintenance decision task may suggest to perform project analyses to improve the coverage of two subject areas requiring the analysis of two distinct sets of projects. Since total effort expenditures for maintenance are limited in a given organization, not all project analyses can be performed. Therefore, during the organization of maintenance, it has to be decided which of the subject areas is to be improved (or whether both subject areas are improved only "a little bit").
- The task *maintain experience cases* has the objective of keeping the experience cases (i.e., the experience stored in the EB) attractive, i.e., useful, usable, and being used

(Feldmann et al. 2000). For this purpose, new experience has to be recorded as experience cases, existing experience cases have to be updated, and obsolete experience cases have to be forgotten (Althoff et al. 1999a). For example, project analyses may be performed to record new project, guideline, and observation cases (see Figure 2).

- The task *maintain EB infrastructure* deals with changes to the EB infrastructure, i.e., conceptual model and implementation as well as the methods, techniques, and tools for recording, updating, and forgetting experience. Such changes can lead to further changes to the experience cases (Tautz 2000; Althoff et al. 1999a). For example, adding a new subject area to the EB requires the extension of the conceptual model and the definition of the respective procedures for recording, updating, and forgetting experience.
- The objective of *learning about maintenance* is to continuously improve EB maintenance itself. This is mainly done by improving the set of maintenance guidelines by developing new and updating existing maintenance guidelines. For example, during the performance of project analyses, the effort can be collected. This effort is then fed back to allow a more precise effort estimation the next time project analyses are to be performed to improve the coverage of a subject area.

To make the performance of these maintenance tasks a systematic effort and support these by methods, techniques, and tools, the tasks have to be assigned to roles that are responsible for performing the tasks, and the tasks have to be operationalized to improve guidance for the roles. This is the subject of the next two sections.

4. ROLES AND RESPONSIBILITIES

When an EB is deployed, it is always embedded in its organizational context: the EF. The EF (as an organizational unit) has to perform a number of tasks (including the maintenance tasks). These tasks are assigned to roles, which are performed by the EF staff. We first introduce a set of EF roles, which form a consolidated model of roles that can be tailored to specific constraints of a concrete industrial setting [according to an extended version of Althoff et al. (1999a)[5]], and then describe for which maintenance tasks they are responsible.

4.1. A Set of EF Roles

The *EF manager* defines strategic goals, initiates improvement programs, and acquires and manages resources. The *experience manager* determines the structure and contents of the EB (e.g., conceptual model) and controls its quality. The *project supporter* is mainly responsible for recording new experience and supporting the project teams in using the existing experience. The *experience engineer* is responsible for packaging and analyzing existing experience. Together with the experience manager, he or she identifies new reuse requirements and acquires experience cases according to them. He or she analyzes the EB to search for improvement potential. The *librarian* is responsible for technical tasks such as the creation and technical maintenance of the EB. He or she stores and publishes experience cases.

[5]In contrast to Althoff et al. (1999a) extended version, we separate the role of the manager into two roles: the EF manager and the experience manager.

4.2. The Maintenance Responsibilities of the EF Roles

The tasks related to maintenance (which were introduced in Section 3) are assigned to the roles as follows:

- The task "analyze EB and its usage" is performed by EF manager and experience manager together. The EF manager provides the business goals and the high-level success criteria. The experience manager is responsible for refining the business goals and high-level success criteria into an operational evaluation program, which exactly defines the measurement activities in the context of the evaluation. EF manager and experience manager might want to hire an external EB evaluation specialist as an independent controller from the outside to improve the evaluation results as well as their credibility. Experience engineers and stakeholders can participate in the evaluation where appropriate and possible.
- The tasks of "organizing and making decisions about EB maintenance" (i.e., the what, when, and who of EB maintenance) are performed by the experience manager in accordance with the general guidelines for human resource issues and maintenance prioritization policies as provided by the EF manager.
- The group of "maintain experience cases" tasks are performed by the experience engineers.
- The contents-specific aspects of the task "maintain EB infrastructure" are performed by the experience manager. He or she also decides about local changes (e.g., adding an attribute, extending a type, changing a recording method). Global changes (e.g., adding a new subject area, adding/removing a particular type of experience cases) involve the EF manager in the decision making. This especially includes decisions that have consequences that cross the border of the area of responsibility of an experience manager. The maintenance of the tools underlying the technical infrastructure is performed by the librarian.
- Tools also perform activities automatically that otherwise would require a librarian in the role set as outlined above.

4.3. Deploying EF Roles in Practice

The actual deployment of an EF in an organization can grow incrementally. In the beginning, one person can perform (almost) all EF roles. As the EB grows, more people are required who can be assigned to the different roles (e.g., one experience manager and several experience engineers per subject area). Additionally, the capabilities of the EF staff members regarding the tasks in the EF have to be considered. Their capabilities can lead to role definitions with slightly different responsibilities.

5. SUPPORT FOR MAINTENANCE TASKS BY THE EMSIG FRAMEWORK

To improve guidance and support for the EF staff, the EMSIG[6] framework (Figure 4) operationalizes the basic ideas. Maintenance is triggered and justified by systematically conducting and exploiting evaluation. Maintenance itself is improved systematically by recording and using experience gained during maintenance.

[6]EMSIG = evaluation and maintenance of software engineering repositories.

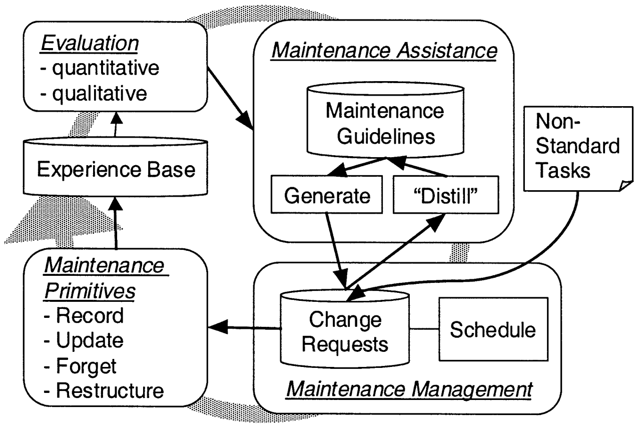

FIGURE 4. The EMSIG framework for maintaining and evaluating an EB.

In the following we first give an overview on the framework and illustrate the framework with a scenario. Then we describe in detail the components and their relationships, how they support the maintenance tasks that are performed by the EF roles, and how particular methods, techniques, and tools from artificial intelligence, knowledge engineering, and software engineering can be used within the framework.

5.1. Overview on the Framework

The EMSIG framework is depicted in Figure 4. The components and relationships are oriented on the tasks on the maintenance loop for performing EB maintenance (see Figure 3) and embed the feedback loop for learning about maintenance.[7]

The *evaluation component* supports the task "analyze the EB and its usage" and thus is responsible for the quality and value issues and deals with the "why" of maintenance. The results of these analyses provide the basis and input for making maintenance decisions. The *maintenance assistance component* supports the decision-making task by exploiting the evaluation in order to propose change requests (i.e., basic maintenance activities to be done). This deals mainly with knowledge issues and the "what" of maintenance ("what" to do for "what" knowledge/experience) and has to consider the "why" (justification from evaluation in the form of expected benefits versus expected maintenance effort). To support the task of learning about EB maintenance, typical tasks or patterns of maintenance activities are identified and captured ("distill maintenance guidelines"). These maintenance guidelines can be used for generating change requests automatically. With this, the feedback loop is embedded into the framework. The *maintenance management component* supports the tasks of organizing maintenance and thus is responsible for handling the change requests in an appropriate order. When a change request is executed, the *maintenance primitives component* provides the actual method,[8] technique, and/or tool(s) to perform the basic maintenance activities on the EB as demanded by the change request.

[7]Note that, actually, the EB system itself and its tools are also an element of the framework. The choice of the EB tools also has an impact on the tools that are used for the evaluation component and the learning component. However, methods usually have a generic part that is applicable to more than one tool.

[8]Note that a method or technique can consist of parts to be carried out manually and parts automated by tools.

"Merging Project Process Descriptions with Lessons Learned"

Trigger:

- Number of lessons learned attached to a project process description is more than 20

Action:

- Decide which of the lessons learned should be integrated into the project process description. (-> Technique "Decision on 'aggregate lessons learned'")
- Decide if the process description should be split into subprocesses. (-> Technique "Decision on 'rearrange process description and lessons learned'")
- According to decision: Split process description and re-assign lessons learned to subprocesses. (-> Technique "Split process description and re-assign lessons learned")
- Aggregate process description(s) and lessons learned. (-> Technique "Aggregate project process description and lessons learned")

Expected benefits:

- The description is more comprehensive and easier to understand.

FIGURE 5. Example maintenance guideline (for scenario A).

5.2. Example Scenarios

We illustrate the framework with a simplified scenario that shows how the components of the framework work together. For this purpose, we first outline a state of an example EB and of its maintenance according to the framework. To keep this simple and easy to understand, this description of the state is restricted to the items relevant for the scenario. Based on the state, we show how two "changes" run through the components.

The Status of the EB and Its Maintenance. The EB of a software development company contains (among many other items) descriptions of the relevant project processes in several variations and lessons learned that are related to these project processes (e.g., see Figure 2 regarding business processes and attached lessons learned in the form of guidelines, observations, and problems). In contrast to Figure 2, 19 lessons learned are attached to the process description "Execute Project."

A number of maintenance guidelines have been identified and defined for the example EB. For the purpose of this scenario, we focus on two of these maintenance guidelines (Figures 5 and 6):

- One maintenance guideline (in Figure 5) aims at keeping the project process descriptions with its attached lessons learned in a state in which they are (still)

"Project Analysis Interviews for Subject Area X"

Trigger:

- At the end of each 10th small project in subject area X, a project analysis interview is performed. ("small" is defined as "less than 1 person month")

Action:

- Make appointment with project members for project analysis interview.
- Conduct project analysis interview using project analysis questionnaire (-> Technique "Project Analysis Interview")
- Qualify and split answers to questionnaire from project analysis (-> Technique "Record Project Analysis Results")

Expected benefits:

- The EB is filled with new experience.

FIGURE 6. Example maintenance guideline (for scenario B).

comprehensible. The underlying assumption is that a process description with too many lessons learned attached is hard to comprehend and use. Thus, this maintenance guideline monitors the number of lessons learned attached to a project process description and notifies the responsible EF staff member if the number of attached lessons learned becomes too high.

- Another maintenance guideline (in Figure 6) helps to identify the need for experience acquisition and notifies the EF staff, respectively. For this EB, experience is collected in so-called project analysis interviews with the project members. Such a project analysis interview takes place at the end of a project. The project analysis interview is conducted using a standardized questionnaire that is filled out during the interview. The answers to the project analysis questionnaire are split into separate experience cases, qualified (i.e., analyzed regarding their reuse potential, reviewed with respect to their quality, and some minor fixes performed if necessary), and then stored in the experience base [see Tautz et al. (2000) for details on project analysis].

  In this example, not all small projects in subject area *X* are subject to project analysis. The reason is that the effort for conducting a project analysis is high for small projects in comparison to the total effort spent in a small project. To keep maintenance effort reasonable, project analyses are conducted only after each tenth small project.

Scenario A. The small project CENT in the subject area *X* ends. Since no project analysis was performed for the nine small projects before CENT, the maintenance guideline "Project Analysis Interviews for Subject Area *X*" (Figure 6) fires, and a change request for performing the project analysis is generated. The experience manager (who is responsible for subject area *X*) assigns the change request to an experience engineer, who performs the project analysis to collect the experience gained in the project. This analysis leads to 15 experience cases, which are recorded by the experience engineer. Among these cases are two new guidelines that are related to the project process "Execute Project."

The evaluation component, which also monitors the measures needed by the triggers of the maintenance guidelines, finds out that now 21 lessons learned are attached to the description of the project process "Execute Project." This information is fed to the maintenance assistance component that recognizes that the trigger of the maintenance guideline "Merging Project Process Descriptions with Lessons Learned" (Figure 5) became true. Thus this maintenance guideline is fired, and the respective change request for the process description "Execute Project" is generated and stored in the CRQ base. The experience manager prioritizes the new change request and assigns an experience engineer who is responsible for "performing" the change request. This experience engineer finds the change request in his or her to-do list. As soon as he or she has the time for doing the change, he or she fetches the actions that are described in the change request (according to the guideline). That is, he or she gets the description of the techniques on making the necessary decisions. In this case, he or she decides to aggregate by integrating five of the lessons learned into the description of the project process "Execute Project." Thus he or she gets the description of the technique for aggregating a process description with (some of) its lessons learned and does the aggregation with the help of the EB tools.

Scenario B. In the definition of an evaluation program for the EB, it was figured out that the coverage of a subject area is considered by the stakeholders of the EB to be a factor that has high impact on the utility and value of the EB. Thus the evaluation component monitors the coverage for each subject area and checks if it remains above a certain level that was decided on by the stakeholders.

In this monitoring of the coverage, the evaluation component finds out that the coverage of subject area X is too low. The reason is that only 10 percent of the small projects in the subject area X are interviewed to collect new experience cases. The experience manager (who is responsible for subject area X) suggests to the EF manager to increase the coverage, and the EF manager agrees. Thus the experience manager changes the respective maintenance guideline (Figure 6) so that more project analysis interviews for this subject area are requested.

5.3. Evaluation Component

The *evaluation component* supports the task “analyze EB and its usage.” Source and focus of the evaluation determine which methods and techniques can be applied in the context of the evaluation component:

- For evaluation focused on value and quality, goal-oriented measurement with the goal-question-metric (GQM) technique is a suitable method, which has been applied successfully to EBs (Nick et al. 1999).

  GQM is actually a framework itself; i.e., it provides the organizational context and management of an evaluation program to facilitate evaluation. Stakeholders can be involved in the definition of the evaluation program based on the business needs/goals for the EB (i.e., the objectives and success criteria identified by the stakeholders during EB buildup) as well as in the interpretation of the measurement data with respect to the goals. For the actual quality models, measures, data collection, and analysis methods and tools, either new ones are developed using GQM or existing ones are plugged in as needed and appropriate [e.g., a competence model from CBR theory (Smyth and McKenna 1998) for analyzing the coverage of a subject area]. The “never ending” iteration of GQM with refinements and rollouts as well as collecting and improving quality models, etc., leads to a learning spiral: We learn more and more about evaluation and, thus, improve the evaluation itself.

  For more in-depth information on GQM for EBs, the reader is referred to Nick et al. (1999).
- User feedback is another source for identifying improvement needs. Part of the organizational memory improvement (OMI) method (Althoff et al. 1999b) describes when we can get which kind of feedback from the users during search for, retrieval, and usage of artifacts from an organizational memory or EB and for what kind of improvement this feedback can be used.

5.4. Maintenance Assistance Component

The *maintenance assistance component* supports the tasks “decide about EB maintenance” and “learn about EB maintenance.” For decision making, different kinds of support are required for standard and nonstandard tasks.

For *standard tasks* (i.e., well-known recurring and/or frequent activities), CRQs are generated from so-called maintenance guidelines, which describe in a structured manner typical tasks or patterns of maintenance activities and the conditions for their instantiation. In spirit, they are related to the CBM policies of Leake and Wilson (1998).

The structure of a maintenance guideline is as follows: A "trigger" states the condition for "firing" the guideline (e.g., number of lessons learned attached to a process description is more than 50). These conditions can refer to evaluation results as well as EB schema and contents. The "expected benefits" and additional scenarios help justify the instantiation of the guideline as CRQ (e.g., by cost-benefit issues, quality improvements, the importance of a scenario) and provide at least hints for assigning a priority or importance level to the CRQ. The "action" describes what has to be done when a guideline is triggered (i.e., which knowledge/experience item has to be maintained and which methods, techniques, and/or tools from the maintenance primitive component are used). Dependencies between guidelines lead to the generation of additional CRQs based on the dependent guidelines. The responsible role can be stated to increase flexibility instead of a fixed assignment of experience case changes to the experience engineer and schema changes to the experience manager.

To enable the automatic generation of CRQs, tool support is essential for the maintenance guidelines. For firing guidelines to generate CRQs, it is necessary to have a formal description of at least (a part of) the trigger of the guideline (e.g., trigger at the end of each month). If a guideline refers to generic items from the EB (e.g., "process descriptions") or to a generic configuration of items, it is necessary to generate a separate CRQ for each single experience case or configuration that is affected.

Nonstandard tasks are mainly derived from decisions made, for example, during evaluation or management decisions (e.g., adding new core competencies). For each of them, a change request is generated manually and stored in the CRQ base.

To support the task of *learning about maintenance,* typical tasks or patterns of maintenance activities are identified and captured ("distill maintenance guidelines"). Maintenance guidelines can be *distilled* from many sources: change requests for nonstandard tasks, experience from the application of guidelines via change requests, evaluation, reuse scenarios, etc. New or refined maintenance guidelines are also "distilled" when dependencies are recognized. This distilling is another point for plugging in artificial intelligence tools. For example, such tools could identify CRQs that are related and/or patch the formal parts of a guideline for a certain context using ripple-down rules (Preston et al. 1993).

5.5. Maintenance Management Component

The *maintenance management component* supports the organization task and thus is responsible for managing the change requests and handling them in an appropriate order.

The *change requests* (CRQs) tell the EF staff which maintenance activities have to be performed for which cases and when. The maintenance activities are described either by the corresponding guideline (if the CRQ refers to a standard task) or simply as more or less structured text (if the CRQ refers to a nonstandard task). The "when" of a CRQ can be a mix of several elements: a deadline (if applicable) and/or a rating of importance (this might be influenced by the expected benefits from the guideline description in some way). Other items might be relevant in certain environments.

What kind of method and tool are appropriate for maintenance management depends at least on the size of the experience factory: For a small number of people, a simple solution can be appropriate (e.g., an email-based approach where the CRQs are sent by email to the EF staff); for a larger number of people (as in a "traditional" factory), more advanced human resources management would be necessary.

5.6. Maintenance Primitives Component

When a change request is executed, the *maintenance primitives component* provides the actual method, technique, and/or tool(s) to perform the basic maintenance activities on the EB as demanded by the change request. These aim at two areas/levels: (1) At the knowledge/contents level, this component provides methods, techniques, and tools for recording new knowledge and experience, for updating, aggregating, and forgetting experience cases, as well as for restructuring the EB. These tasks require domain-specific "procedures" (Tautz 2000). An example of a method for recording and packaging well-structured lessons learned can be found in Birk and Tautz (1998). Since updating experience cases can take some time, it is useful to have the last validated version of each of these experience cases available for queries. This is subject to the life-cycle model for experience cases in (Minor and Hanft 2000). (2) At the technical level (i.e., representation and implementation), this component deals with methods for keeping the EB consistent. How this can be done technically for CBR systems is described in Heister and Wilke (1998). Most of the techniques described in this article have been implemented in the commercial CBR tool CBR-Works. In the future it also would be desirable to acquire and/or extract knowledge automatically, e.g., as described in Patterson et al. (1998).

6. INTEGRATED CBR-BASED EXPERIENCE BASE IMPLEMENTATION

For the practical application of the framework, we give an overview on an integrated technical solution for an EB that is maintained using our framework. The integrated solution extends our solution (Section 2.1; Althoff et al. 1999a), which consists of a common conceptual model and a scalable EB architecture. The implementation of EMSIG requires the extension of both the conceptual model for representing the maintenance knowledge and the architecture for adding new functionality required for maintenance (see Section 6.1). Afterwards, we outline the use of EMSIG in COIN (see Section 6.2).

6.1. General Implementation Issues

The *common conceptual model* (Figure 7) integrates experience cases and cases with maintenance information at the conceptual level. For each of the components of the framework, a new top-level case concept is added.[9] Figure 7 depicts the types of cases at the top level, the relationships among the new concepts, and the relationships to the experience case concepts (i.e., EB schema) and their respective instances (i.e., actual experience cases). The relationships reflect the structure of the maintenance guidelines and change requests from the EMSIG framework (see Section 5). In addition, the model includes the relation of a change request to the maintenance guideline it was generated from, which is necessary for tracking, regardless of whether a change request has been generated or not.

The top-level case concepts can be refined and inherited as necessary. For example, all the concepts from Figure 2 ("business process," "guideline," "problem," "observation," "project," "customer," etc.) are subconcepts of "experience case." The concept "measurement data" is usually refined to "measurement data on query results" (e.g., textual feedback on the whole query result) and "measurement data on cases in query

[9]With CBR-Works, all top-level case concepts are subconcepts of a root concept "Case."

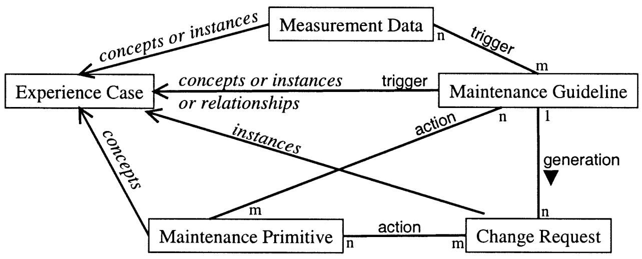

FIGURE 7. Framework for the conceptual model of an experience base including maintenance cases.

results" [e.g., textual feedback on the whole query result and perceived usefulness of a retrieved case (Nick et al. 1999)].

The *EB architecture* for the technical implementation of the EB is based on the common conceptual model. For the implementation of the EB itself, structured CBR has demonstrated its effectiveness and efficiency in an industrial-strength case study using INTERESTS with CBR-Works (Tautz 2000; Tautz and Althoff 2000). Because the maintenance knowledge also shows a structured nature, we physically integrate schema and cases for maintenance knowledge with the actual experience. Thus we can still use CBR-Works or orenge[10] as well as the other tools on top (in particular, EB server and general-purpose browser application framework).

The components of the EMSIG framework require (1) additional interactive tools and (2) functionality for recurring tasks. The former requires tools that are invoked by the user; the latter requires tools that run recurrently. Because some components have both aspects, separate tools must be developed for these. The EMSIG tools fit into the provisioned slots in our scalable architecture (Althoff et al. 1999a), which are on top of the EB server, and complement INTERESTS (see Section 2.1).

There are two interactive tools for maintenance: The *maintenance manager* provides the following features: prioritization and assignment of CRQs, which is simply a case modification of CRQ cases with undefined assignment, and overviews on CRQs by status and role or person, which are used as to-do lists and for work status monitoring. These are sorted by priority, which reflects the "when" of a CRQ. Because the priority can be composed of several elements (see Figure 8), a similarity measure implements the CRQ priority composition policy and allows sorting the CRQs by the composed priority using their similarity to the query case, which has the priority attributes set to their most urgent values (e.g., importance very high, deadline now). The *maintenance assistant* allows browsing and editing maintenance guidelines, which supports the manually performed "distill." Because both maintenance manager and maintenance assistant require only standard case retrieval and manipulation features, they could be developed rapidly using the general-purpose browser application framework. The maintenance primitive descriptions are edited with the general-purpose browser. The more than 100 tasks descriptions from DISER (Tautz 2000) are used as an initial set of maintenance primitive descriptions.

The remaining tools have no interactive user interface and run recurrently. The *maintenance assistance demon* generates change requests based on measurement data

[10]To use another CBR tool, it would be necessary for this tool to satisfiy the requirements from Tautz (2000). An interface that is different from the interfaces of CBR-Works or orenge (i.e., CQL or XML) would require changes to the EB server implementation.

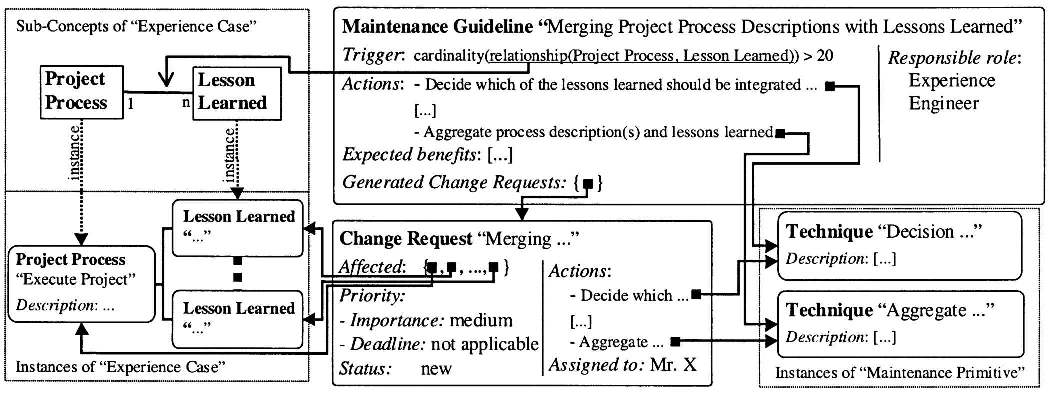
Sub-Concepts of "Experience Case"
Project Process
Lesson Learned
1
n
instance
instance
Lesson Learned "..."
Project Process "Execute Project" Description: ...
Lesson Learned "..."
Instances of "Experience Case"
Maintenance Guideline "Merging Project Process Descriptions with Lessons Learned"
Trigger: cardinality(relationship(Project Process, Lesson Learned)) > 20
Actions: - Decide which of the lessons learned should be integrated ...
[...]
- Aggregate process description(s) and lessons learned
Expected benefits: [...]
Generated Change Requests: { }
Responsible role: Experience Engineer
Change Request "Merging ..."
Affected: { , , ..., }
Priority:
- Importance: medium
- Deadline: not applicable
Status: new
Actions:
- Decide which ...
[...]
- Aggregate ...
Assigned to: Mr. X
Technique "Decision ..." Description: [...]
Technique "Aggregate ..." Description: [...]
Instances of "Maintenance Primitive"

FIGURE 8. Example maintenance guidelines (Figure 5) with generated change request—represented in the common conceptual model.

and on the results of EB analyses, which are provided by the *EB analyzer.* The *CRQ analyzer* does the automatic part of "distill" (not yet developed). The automatic generation of new items is followed by optional notification of the responsible role/person by, for example, email.[11]

To perform measurement data analyses without an explicit relation to maintenance guidelines (e.g., GQM-based evaluation), the required measurement data are imported into a standard statistic tool, which is used for the actual analyses.

6.2. Application in COIN: Is EMSIG Effective for COIN?

COIN's two major subject areas, business process descriptions (COIN-IQ) and lessons learned about project management (COIN-EF), are very different with respect to maintenance and therefore have different foci with respect to EMSIG.

For the business process descriptions, the focus is on the maintenance management component, because there is only one part time experience manager for this subject area who has to "supervise" a number of experience engineers creating and updating the business process descriptions. All of them work for COIN only part time or even only from time to time. The reason for this is the policy that process owners are responsible for the maintenance of their business processes. Someone "owns" a process either because it falls into his or her domain of work (e.g., library, procurement, publication service) or into his or her domain of competence (e.g., contracts, project process). This has the effect that there are currently nine process owners and approximately eight designated ghostwriters for the process owners with very little time available. An interview with the responsible experience manager showed that—due to the part-time nature of the job—tool support for maintenance management in general and a reminder feature for "reminding to remind owners and authors" in particular are rated very useful.

For the lessons learned about project management, the focus is on maintenance assistance. Here we want to learn more about maintenance by creating, distilling, exploring, and validating "content oriented" maintenance guidelines. Such maintenance guidelines, for example, describe when to generalize similar lessons learned or when to integrate lessons learned into related process descriptions. Figure 8 shows an example

[11]These demons can be viewed as active CBR components in the sense of Li and Yang (2000).

of such a maintenance guideline together with a related change request for a particular configuration. The example uses the common conceptual model from Section 6.1.

In addition, there is also the benefit of getting more guidance and better support for the organization of the maintenance of the lessons learned because the maintainers of this subject area work only part-time in the EF and thus tend to forget or drop necessary activities if they are not reminded by EMSIG.

7. DISCUSSION ON EMPIRICAL VALIDATION

EB maintenance has to attain a certain goal, which was stated as "to preserve and/or improve the EB value" (see Section 1). Thus empirical validation and experimentation have to investigate if, how, and under which circumstances the particular instantiations of EMSIG or a part of EMSIG (i.e., a particular method, technique, or tool from EMSIG or a particular maintenance guideline) contribute to that goal. This requires (1) monitoring the EB value over time, i.e., operational measures for the EB value, and (2) analyzing the impact of EMSIG or its parts as well as the factors mentioned in Section 1 (different types of knowledge, size of EF staff, etc.) and maybe other factors that have not been identified yet. Due to the number of factors mentioned, it is too expensive and often practically impossible to control them in an industrial-strength environment (Kitchenham et al. 1995). Thus we use case studies on EMSIG for such environments and controlled experiments for very specific studies on parts of EMSIG.

Measuring the EB Value. The theoretically correct EB value would be defined as the value of all the query results less the costs for all queries as well as buildup and maintenance activities. The value of a query result is the value of the retrieved cases in terms of money or effort saved and includes any future use of the retrieved cases that happens without querying the EB.

Since operationalization of the theoretically correct value is very difficult in general, practical measurement uses the monitoring of indicators to identify trends regarding the value of an EB (Feldmann et al. 2000; Nick et al. 1999). Such indicators are usage, utility as perceived by the stakeholders, and usability. These indicators have certain restrictions and problems: Usability only ensures that the EB is used and thus that the EB can have a certain utility. High usage does not guarantee that the users find useful experience cases, but sustained high usage does. No or less usage can have its reasons in insufficient perceived utility or insufficient usability. Measuring sustained usage requires monitoring of usage over a longer period of time. This is not necessary when measuring the perceived utility of the retrieved cases. Such a utility measure also was used for an experiment with COIN (Tautz and Althoff 2000). Therefore, we use two indicators for the EB value: sustained usage of the EB and perceived usefulness of the retrieved cases.[12]

Using the indicator "sustained usage" in practice requires a "calibration" to determine meaningful time-interval sizes for analyzing usage. If the interval size is too small, the number of usages for the interval will rise and fall without real meaning, and thus analysis would be impossible. If the interval size were too large, the study could take very long before we would get meaningful results.

[12]There are also additional benefits from measuring the perceived usefulness (Althoff et al. 1999b); e.g., the similarity measures can be tested and validated for the EB under focus.

EB Maintenance Case Study: Effectiveness and User Acceptance of EMSIG. The main focus of the case study is on the effectiveness of EMSIG. This is combined with an investigation on user acceptance to ensure the usability and usage of EMSIG throughout the study in order to achieve meaningful results.

The "straightforward" goal for the study would be to analyze if using EMSIG has a positive impact on the EB value (measured with the indicators stated above). But a case study with this goal is not possible because of the many factors influencing maintenance and the EB value (Kitchenham et al. 1995). Therefore, for the case study, we relax the goal as follows: Analyze if using a particular instantiation of EMSIG has no negative impact on the sustained usage and perceived usefulness of the EB. Still, the many factors influencing maintenance and the EB value can threaten the study's validity. Long-term studies in particular run the risk of measuring the impact of slow changes of factors not under focus (e.g., changes in the organizational culture) instead of EMSIG's impact. To improve confidence in the causal relationship under focus, the following measures can be taken: Consider interference in the analyses; i.e., partition the data in case of changes with EMSIG or maintenance without EMSIG. Check that not only a single part of EMSIG is used, because in this case we would not show the impact of EMSIG as a whole. If the EF staff refuses to use EMSIG, it cannot have an impact on the EB value, and data collected in that period of time are not considered meaningful with respect to the goal of the study. In addition, the reasons should be identified, e.g., with simple interviews or detailed usability studies, and the respective problems fixed in order to get results related to EMSIG.

We are going to conduct such a case study for COIN and hopefully replicate the study for further EBs using EMSIG. Such a case study requires monitoring the usage of EMSIG and the usage and perceived utility of the EB as well. We combine this with an investigation into the acceptance of the parts of EMSIG by the COIN team members in order to identify and fix usability and functionality problems early.

Nevertheless, the study cannot investigate if one part of EMSIG has a positive effect and another part has a negative effect. If there are such concerns, a controlled experiment can help to analyze such a particular presumption.

Controlled Experiments. Controlled experiments are particularly useful for analyzing the impact of a maintenance guideline, comparing maintenance guidelines (in the same "area"), or analyzing the impact of parameters of a maintenance guideline (e.g., the number of lessons learned for merging in Figure 8). Comparing the impact of having multiple guidelines and respective decision support for selection with the impact of single maintenance guidelines would show that EMSIG's approach is more effective than a single maintenance guideline/strategy. For the sake of higher external validity, an industrial-strength EB should be used for such experiments (e.g., COIN).

For such experiments, the indicator "sustained usage" is not applicable because running such an experiment over a longer period of time can threaten the internal validity of the experiment. Thus perceived usefulness should be used, as it was done successfully in an experiment with COIN in Tautz (2000).

8. CONCLUSION (SUMMARY, FUTURE WORK)

The value of a corporate information system tends to degrade with time, be it by external impacts on the organization's environment or by changes within an organization (e.g., the development of a new product). This is particularly true if exemplary knowledge is stored in the information system, as is typically done in CBR systems because such knowledge is gained almost continuously in daily work. To preserve and/or improve the value of a CBR system for its stakeholders, the system must be maintained on a continuous basis.

This article presents an evaluation and maintenance methodology for systems storing all kinds of experience (called *experience bases*). The methodology has its roots in the experience factory (EF) concept (Basili et al. 1994). The EF is an organizational unit separate from those units responsible for daily business. Its objective is to improve (and thus to maintain) the experience of an organization in a systematic fashion by collecting and storing experience explicitly and keeping it attractive for and being used by its intended users.

The evaluation and maintenance methodology is based on the ideas of driving experience base (EB) maintenance by systematically conducting and exploiting evaluation and of systematically performing maintenance itself by recording and using experience gained during maintenance in the form of special maintenance guidelines.

The methodology consists of tasks to be performed, roles needed to evaluate and maintain an experience base, a supporting technical framework, and a (generic) CBR-based implementation of the framework. The tasks can be separated into two loops: the maintenance loop (analyzing the EB and its value, deciding about maintenance, organizing maintenance, and performing basic maintenance activities) and the feedback loop, which gains maintenance experience (in the form of maintenance guidelines) from the analysis and basic maintenance activities and feeds it back to the decision-making and organizing tasks. For each of the tasks, roles are assigned as they are typically encountered in EF setups. The framework (called EMSIG), which corresponds to the tasks, is the basis for the definition of supporting methods, techniques, and tools. For the practical application, we have demonstrated that many of EMSIG's features can be implemented with a standard CBR tool such as CBR-Works or orenge without much effort. Thus CBR is a suitable technology not only for implementing an EB but also for managing and conducting EB maintenance. Nevertheless, the firing of maintenance guidelines, which is the core element of the approach, requires additional support by an active component (Li and Yang 2000).

We are aiming at making the methodology applicable in industrial-strength projects. We do so by instantiating the framework components with industrial-strength methods, techniques, and tools. For example, the evaluation component can be instantiated by the industrial-strength GQM technique (Nick et al. 1999) to systematically derive maintenance triggers (telling when to start a maintenance loop) from business goals. The methodology has been developed based on evaluations of existing experience bases (Nick et al. 1999; Feldmann et al. 2000) and maintenance problems encountered while running industrial-strength experience base projects (Althoff et al. 1999a). The tools for the EMSIG framework will be used for our corporate information network (COIN).

Having additional maintenance knowledge for an EB leads to the so-called recursive maintenance problem, i.e., the question of how to maintain the maintenance knowledge. According to Menzies (1998), this can only be solved by keeping the maintenance knowledge simpler than the original EB. Obviously, this holds for the structure of the maintenance guidelines and change requests (see Figure 7) compared with the more complex structure of a typical EB (see Figure 2). Because of this and COIN's practical needs and expected benefits (see Section 6.2), we see EMSIG as an effective solution to the maintenance problem. We are going to empirically validate this with industrial-strength case studies and controlled experiments as outlined and discussed in Section 7.

ACKNOWLEDGMENTS

We thank Frank Bomarius and Björn Decker for the fruitful discussions on the topic. We thank Sonnhild Namingha for reviewing an earlier version of this article.

REFERENCES

AAMODT, A., and E. PLAZA. 1994. Case-based reasoning: Foundational issues, methodological variations, and system approaches. AICom, Artificial Intelligence Communications, **7**(1):39–59.

ALTHOFF, K.-D., A. BIRK, S. HARTKOPF, W. MÜLLER, M. NICK, D. SURMANN, and C. TAUTZ. 1999a. Managing software engineering experience for comprehensive reuse. *In* Proceedings of the Eleventh Conference on Software Engineering and Knowledge Engineering. Knowledge Systems Institute, Skokie, IL. Kaiserslautern, Germany, pp. 10–19. An extended version is available as Technical IESE-Report No. 001.99/E, Fraunhofer IESE, Kaiserslautern, Germany.

ALTHOFF, K.-D., M. NICK, and C. TAUTZ. 1999b. Improving organizational memories through user feedback. *In* Workshop on Learning Software Organisations at SEKE'99. Kaiserslautern, Germany.

BASILI, V. R., G. CALDIERA, and H. D. ROMBACH. 1994. Experience factory. *In* Encyclopedia of Software Engineering. *Edited by* J. J. Marciniak. John Wiley & Sons, New York, pp. 469–476.

BASILI, V. R., and H. D. ROMBACH. 1988. The TAME Project: Towards improvement-oriented software environments. IEEE Transactions on Software Engineering, **SE-14**(6):758–773.

BIRK, A., and C. TAUTZ. 1998. Knowledge management of software engineering lessons learned. *In* Proceedings of the Tenth Conference on Software Engineering and Knowledge Engineering. Knowledge Systems Institute, Skokie, IL.

FELDMANN, R. L., M. NICK, and M. FREY. 2000. Towards industrial-strength measurement programs for reuse and experience repository systems. *In* Second International Workshop on Learning Software Organizations (LSO 2000), Oulu, Finland.

FERRARIO, M. A., and B. SMYTH. 2000. A user-driven, distributed maintenance strategy for large-scale case-based reasoning systems. *In* ECAI Workshop Notes: Flexible Strategies for Maintaining Knowledge Containers, Berlin, Germany. *Edited by* M. Minor. BSO Marketing Gesellschaft, Berlin.

HEISTER, F., and W. WILKE. 1998. An architecture for maintaining case-based reasoning systems. *In* Advances in Case-Based Reasoning: Proceedings of the Fourth European Workshop on Case-Based Reasoning. *Edited by* B. Smyth and P. Cunningham. Springer-Verlag, Berlin, pp. 221–232.

KITCHENHAM, B., L. PICKARD, and S. L. PFLEEGER. 1995. Case studies for method and tool evaluation. IEEE Software, **12**(4):52–62.

LEAKE, D. B., and D. C. WILSON. 1998. Categorizing case-base maintenance: Dimensions and directions. *In* Advances in Case-Based Reasoning: Proceedings of the Fourth European Workshop on Case-Based Reasoning. *Edited by* B. Smith and P. Cunningham. Springer-Verlag, Berlin, pp. 196–207.

LI, S., and Q. YANG. 2000. Activating case-based reasoning with active databases. *In* Advances in Case-Based Reasoning: Proceedings of the Fifth European Workshop on Case-Based Reasoning. *Edited by* E. Blanzieri, and L. Portinale. Springer-Verlag, Berlin, pp. 3–14.

MENZIES, T. 1998. Knowledge maintenance: The state of the art. The Knowledge Engineering Review.

MINOR, M., ed. 2000. ECAI Workshop Notes: Flexible Strategies for Maintaining Knowledge Containers. Berlin, Germany. Springer-Verlag, Berlin.

MINOR, M., and A. HANFT. 2000. Corporate knowledge editing with a life cycle model. *In* 8th German Workshop on Case-Based Reasoning, Laemmerbuckel, Germany. Daimler-Chrysler, Ulm, Germany.

NICK, M., and K.-D. ALTHOFF. 2000. The challenge of supporting repository-based continuous learning with systematic evaluation and maintenance. *In* Third Workshop on Intelligent Software Engineering (WISE3) at ICSE'2000. *Edited by* T. Menzies. University of Limerick, Ireland.

NICK, M., K.-D. ALTHOFF, and C. TAUTZ. 1999. Facilitating the practical evaluation of organizational memories using the goal-question- metric technique. *In* Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management, Banff, Alberta, Canada. SRDG Publications, Calgary, AB.

PATTERSON, D., W. DUBITZKY, S. ANAND, and J. HUGHES. 1998. On the automation of case base development from large databases. *In* AAAI 1998 Workshop on Case-Based Reasoning Integrations. AAAI Press, Menlo Park, CA, pp. 126–130.

PRESTON, P., G. EDWARDS, and P. COMPTON. 1993. A 1600 rule expert system without knowledge engineers. *In* Second World Congress on Expert Systems. *Edited by* J. Leibowitz. Pergamon, Lisbon, Portugal.

REINARTZ, T., I. IGLEZAKIS, and T. ROTH-BERGHOFER. 2000. On quality measures for case base maintenance. *In* Advances in Case-Based Reasoning: Proceedings of the Fifth European Workshop on Case-Based Reasoning. *Edited by* E. Blanzieri and L. Portinale. Springer-Verlag, Berlin, pp. 247–259.

RICHTER, M. M. 1998. Introduction. *In* Case-Based Reasoning Technologies: From Foundations to Applications. *Edited by* M. Lenz, B. Bartsch-Spörl, H.-D. Burkhard, and S. Wess. Springer-Verlag, Berlin, pp. 1–15.

SCHULZ, S. 1999. CBR-Works: A state-of-the-art shell for case-based application building. *In* 7th German Workshop on Case-Based Reasoning, Wuerzburg, Germany. Centre for Learning Systems and Applications, University of Kaiserslautern, Germany.

SMYTH, B., and M. T. KEANE. 1995. Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. *In* Proceedings of the International Joint Conference on Artificial Intelligence. Morgan Kaufmann, Los Angeles, pp. 377–382.

SMYTH, B., and E. MCKENNA. 1998. Modelling the competence of case-bases. *In* Advances in Case-Based Reasoning: Proceedings of the Fourth European Workshop on Case-Based Reasoning. *Edited by* B. Smyth and P. Cunningham. Springer-Verlag, Berlin, pp. 196–207.

TAUTZ, C. 2000. Customizing software engineering experience management systems to organizational needs. Ph.D. thesis, University of Kaiserslautern, Germany.

TAUTZ, C., and K.-D. ALTHOFF. 2000. A case study on engineering ontologies and related processes for sharing software engineering experience. *In* Conference on Software Engineering and Knowledge Engineering (SEKE 2000). Knowledge Systems Institute, Skokie, IL.

TAUTZ, C., K.-D. ALTHOFF, and M. NICK. 2000. A case-based reasoning approach for managing qualitative experience. *In* AAAI-00 Workshop on Intelligent Lessons Learned Systems. AAAI Press, Menlo Park, CA.

VAN HEIJST, G., R. VAN DER SPECK, and E. KRUIZINGA. 1996. Organizing corporate memories. *In* Proceedings of the Tenth Banff Knowledge Acquisition for Knowledge-Based Systems Workshop. SRDG Publications, Calgary, AB.

ZHU, J., and Q. YANG. 1999. Remembering to add: Competence-preserving case addition policies for case base maintenance. *In* Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-99). Morgan Kaufmann, Los Angeles, CA.