Fraunhofer Institut
Experimentelles
Software Engineering

Engineering Experience Base Maintenance Knowledge

Authors:
Markus Nick
Klaus-Dieter Althoff

IESE-Report No. 018.01/E
Version 1.0
March 17, 2001

A publication by Fraunhofer IESE

Fraunhofer IESE is an institute of the Fraunhofer Gesellschaft.
The institute transfers innovative software development techniques, methods and tools into industrial practice, assists companies in building software competencies customized to their needs, and helps them to establish a competetive market position.

Fraunhofer IESE is directed by
Prof. Dr. Dieter Rombach
Sauerwiesen 6
D-67661 Kaiserslautern

Abstract

The value of a corporate information system such as an experience base tends to degrade with time. To keep the value of such a system, maintenance is an essential. Maintenance should not simply happen ad-hoc but systematically and based on specific maintenance knowledge. As a jump-start for maintenance, maintenance knowledge should be available right from the start of continuous operation. This paper describes how to systematically develop ("to engineer") such maintenance knowledge during buildup of the corporate information system.

Keywords: Maintenance, Maintenance Knowledge, Experience Base, Experience Factory

Table of Contents

1 Introduction

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 (experience, case-specific knowledge) is stored in the information system, as is typically done in experience bases (EBs), lessons learned systems, best practice databases, or case-based reasoning (CBR) systems, because such knowledge is gained almost continuously in daily work [42, 47, 16, 3, 46]. In the recent past the new field "Experience Management" [45; 14] appears to establish itself. Experience Management deals with all the relevant research and development issues for this kind of information systems. The ingredients of experience management stem from different scientific fields, among others experience factory/learning software organization, case-based reasoning, and knowledge management.

In Software Engineering, the *experience factory* approach was introduced in the late eighties [13, 4, 6, 41]. It explicitly deals with continuous (organizational) learning from experience. In the areas of Cognitive Science and Artificial Intelligence, *case-based reasoning* emerged in the beginning eighties as a model for human problem solving and learning [43]. In Artificial Intelligence, this led to a focus of knowledge-based systems on experience (experience knowledge, case-specific knowledge) in the late eighties and beginning nineties, mostly in the form of problem-solution cases [11, 8, 1, 36, 10, 4].

In the eighties and nineties, various approaches in economical and social science as well as in business information systems, which explicitly dealt with knowledge as a resource of increasing importance, merged under the notion of *knowledge management* [35, 18, 40, 25]. In spite of the high number of approaches and their heterogeneity, two main categories can be identified [2]: On the one hand, there are process-oriented approaches [21], which base mainly on communication and collaboration, on the other hand, product-oriented approaches, which base on documentation, storage, and reuse of enterprise knowledge [6]. While the former use techniques from computer supported collaborative work and workflow management, the latter build on information technology tools for documenting knowledge: Database systems, repository systems, hypertext systems, document management systems, process modelling systems, knowledge- based systems, case-based reasoning systems, etc. [19].

From a more general perspective it can be stated that product- and process-oriented approaches are still not integrated. Usually they are used independently from each other, or as alternatives. As one exception here, meanwhile a deep - i.e. the cognitive science foundations considering - integration of the approaches of experience factory and case-based reasoning has been achieved [45, 5].

In this paper we deal with experience management approaches. While the term "management" underlines the "process" aspects of such approaches, we subsume the "product" aspects under the notion of experience base systems (or experience bases, case-based reasoning systems, case bases, organizational memory systems, organizational memory, corporate information system). Experience management includes methods for identifying, collecting, documenting, packaging, storing, generalizing, reusing, tailoring, and evaluating of experience (experience packages, cases) based on an EB.

Experience base systems must be maintained on a continuous basis [15, 32]. Such maintenance should not be performed ad-hoc. Instead, a systematic approach is required to ensure "good", well-controlled maintenance. For this purpose, knowledge-focused and technical issues as well as organizational issues have to be considered. To deal with the organizational issues, a dedicated role - e.g., a Chief Information Officer (CIO) - and/or a dedicated organizational unit - e.g., an experience factory (EF) - should be established [15, 6, 12]. To support the actual maintenance of the EB, specific maintenance experience and its conceptual structure has to be included [27, 28, 23, 32].

Maintenance knowledge has been partly discussed in the literature: Leake & Wilson [24] introduced the concept of "maintenance policies" for maintaining the experience base. These "maintenance policies" can be viewed as maintenance knowledge. Rombach [39] discussed principles for the maintenance of cost models, which were based on the dimensions of software maintenance by [44]. Minor & Hanft [29] presented a life-cycle model for test cases and a life-cycle model for lessons learned is available in [7]. McKenna & Smyth [26] presented competence-preserving maintenance strategies for planning tasks.

Another type of knowledge that is related to maintenance, is quality knowledge. Quality knowledge describes how the quality of the EB is defined and how to measure quality as well as the rationale for the quality knowledge [27]. Quality knowledge deals with quality aspects of the EB system as a whole, i.e., the EB's contents and conceptual model as well as retrieval mechanisms and usability of the user interface. An example for content-related quality knowledge is a definition of metrics for the utility or value of single experience packages [34].

In [32], we presented the EMSIG framework and an integrated technical solution operationalizing the (decision) support for the maintenance of an experience base regarding experience packages and conceptual model using specific maintenance and quality knowledge. While the quality knowledge can be acquired and improved using a systematic approach [33, 34], the maintenance knowledge is rather acquired "by chance" during continuous operation (with the exception of maintenance strategies such as competence-preserving case base maintenance strategies for planning tasks [26]). Thus, it might take long to learn the required maintenance knowledge. The problem is that existing methods such as INRECA [15] or DISER[1] [45] only fill the "standard" knowledge containers of CBR/EB systems (vocabulary, cases, similarity measures, adaption [38]) and do not address the acquisition and usage of maintenance and quality knowledge.

This brings us to three open issues that are subject of this paper: (1) How to acquire and develop maintenance knowledge systematically. Our approach derives operational maintenance knowledge from artifacts and information gained during EB buildup and from a knowledge/experience life-cycle model. (2) After the maintenance and quality knowledge has been developed, it has to be integrated into the operational EB system. (3) Such systematic maintenance and quality knowledge acquisition must be integrated into a buildup method.

The development of the required maintenance and quality knowledge during EB buildup ensures that the maintenance and evaluation needs are considered during the development of the EB. This aims at avoiding more expensive future changes due to maintenance needs identified later than possible.

The paper is structured as follows. Section 2 states some assumptions about the EB buildup and evaluation, and gives a glimpse on our in-house experience factory COIN. Section 3 presents our approach to systematically developing maintenance knowledge and illustrates this with examples from COIN and industrial projects. Section 4 describes how to integrate this maintenance and quality knowledge into an EB system using currently available CBR tools such as Orenge from tec:inno/empolis and how to integrate evaluation into the usage of the EB. Section 5 discusses how to integrate the method into an EB buildup methodology. Finally, some conclusions are drawn (Section 6).

1 DISER is a methodology for designing and implementing software engineering repositories

2 Assumption about EB Buildup

During buildup, the following artifacts and information are developed: Objectives (including high-level success criteria) and subject areas of the EB need to be identified. For these, detailed experience reuse and record scenarios are developed. Based on the scenarios, the conceptual model underlying the experience packages is developed. Processes/methods for recording and utilizing experience are defined/selected. A knowledge collection plan describes when which artifact has to be collected how by whom. Finally, the actual technical and non-technical infrastructure must be implemented according to the organization's needs. All these parts and intermediate artifacts/ information should be developed using some methodology such as INRECA or DISER (see [45] 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.

The result of the initial acquisition of quality knowledge (i.e., of the planning phase of an evaluation program) are (1) a measurement plan that describes which measurement data is collected when, how, by whom, and who validates and stores the data; and (2) an evaluation plan that defines when to conduct analyses of the collected measurement data and when to involve users in the interpretation of the analysed data. This should be developed in a systematic manner, e.g., with the goal-oriented Goal-Question-Metric (GQM) method for the measurement plan (see [33, 34] for details). With GQM, the evaluation can be linked to business goals, e.g., from the learning-and- growth part of a balanced scorecard [17, 22]. To jump-start evaluation, GQM allows to reuse and validate existing quality models [33, 9] and quality measures [37]. Still, there are open issues: integration into usage and the integration of lessons learned from the evaluation program into the EB (e.g., guideline/rule about relationships between variation and quality factors). These issues will be addressed in Section 4.

Our in-house experience factory COIN (**Co**rporate **I**nformation **N**etwork) was launched -due to the rapid growth of the IESE- to (a) provide the less experienced people with default processes and guidelines to jump-start them and (b) to facilitate experience sharing among them to build up their expertise more quickly [45, 32]. Since the size of our institute does not allow to talk to all people on a weekly basis, experience sharing on a personal basis does not work. The experience base was developed using the DISER method.

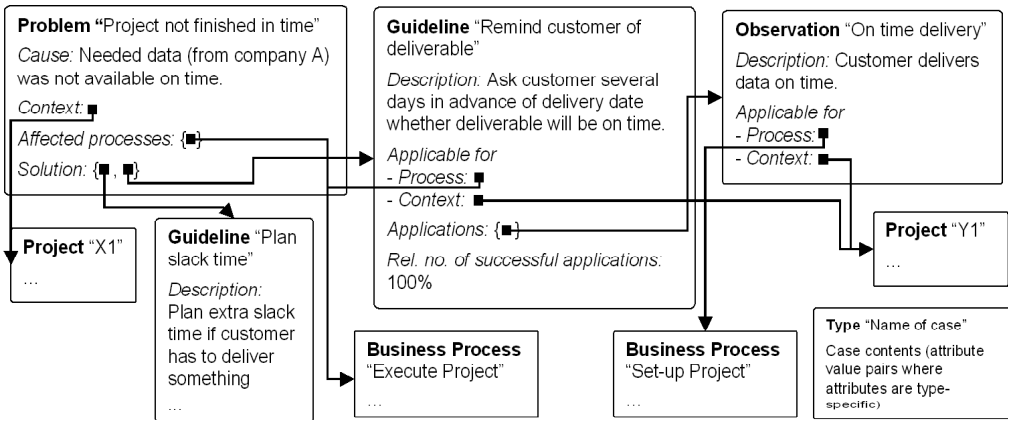

Figure 1 Excerpt of COIN experience base (8 experience packages)

Figure 1 shows an excerpt of COIN's experience base, which will be used in examples in the remainder of this paper. 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 either the results of an application of a guideline or something interesting that has been observed without related experiences.

3 A Method for Engineering Maintenance Knowledge

The principle of engineering maintenance knowledge during buildup (Fig. 2) is to derive operational maintenance knowledge from three major sources: (1) a knowledge/experience life-cycle model, (2) artifacts and information gained during EB buildup, and (3) the measurement plan for the evaluation. From these sources, we derive rather informal maintenance policies and more formal maintenance guidelines. The maintenance policies are further formalized as maintenance guidelines. The maintenance guidelines can be automated using EMSIG's maintenance decision support components (i.e., maintenance assistance and maintenance management component [32]).

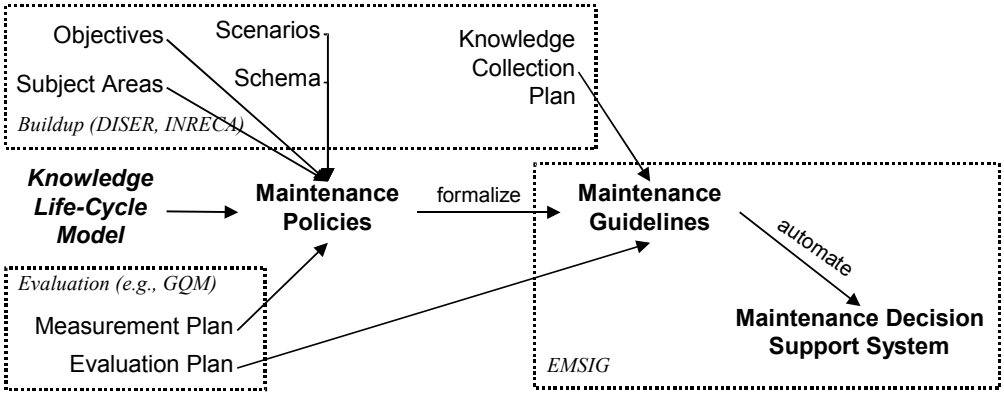

Figure 2 Systematically developing maintenance knowledge and support during EB buildup.

The following sections describe the definition of a knowledge life-cycle model and the deriving of maintenance policies in detail and illustrate this with examples from our in-house experience factory COIN and from industrial projects. Furthermore, the formalization of the maintenance policies into maintenance guidelines and the automation using EMSIG's tools are summarized.

3.1 Defining a Knowledge Life-Cycle Model

A *knowledge/experience life-cycle model* describes the basic idea of how to maintain and improve knowledge and experience over time. Thus, it is the basis of any further refinement of the maintenance process, in particular, with respect to the EB content. The life-cycle model also addresses issues such as validity and degree of maturity of the knowledge and experience stored. All

this is the starting point for a systematic maintenance and improvement of the knowledge and experience.

A generic knowledge life-cycle model is depicted in Fig. 3 [7]. This model has been used for COIN and industrial projects. Explicitly documented knowledge is developed and matured over time: In the beginning, very context-dependent experiences are collected, e.g., from measurement data analyses or expert interviews. Examples for such experiences on very specific issues are the observations, guidelines, and problems in COIN (Fig. 1). By reuse in similar contexts, these experiences are more and more validated. Furthermore, more experiences are collected that are on similar or related issues like the experiences already stored. In addition, these experiences are further aggregated and generalized. When these experiences cover a sufficiently wide area and are mature enough, they can be combine in order to derive a comprehensive best-practice description (e.g., a business process description as in COIN - see Fig. 1).[2] Such a best- practice description is enriched with further experiences, which are integrated into the best-practice description from time to time. This closes the loop.

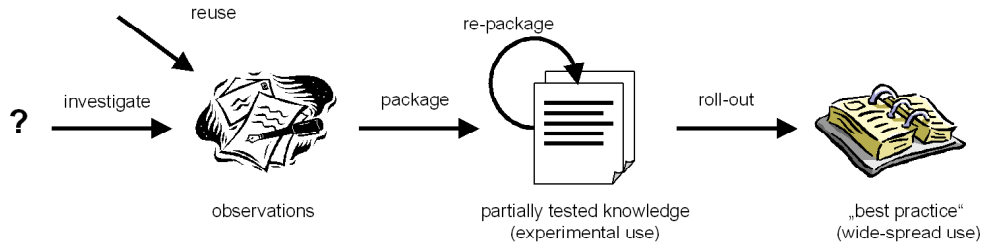

Figure 3 Generic knowledge life-cycle model. [7]

In parallel or triggered by the evolution of the knowledge, the conceptual model of the knowledge is also improved: Knowledge and experience can be more formalized. This leads to defining more attributes and extending existing attribute types.

While the form/type of the knowledge in the life-cycle model (i.e., first sample, lesson learned, best practice) deals with the maturity of the knowledge, more in-depth issues address validity.

Validity describes how general an experience is and how much one can trust the experience to be successfully reused in its anticipated application context. To integrate validity issues into the life-cycle model, an operational definition of validity is required. This definition can be in a qualitative [30] or quantitative manner (COIN).

2 When tacit best practice is available (e.g., knowledge about a business process that has not been made explicit), best-practice descriptions are also elicited directly using suitable knowledge acquisition methods (e.g., business process elicitation & modelling [20]).

If the reuse of experiences can be measured, a quantitative validity can be defined using the number of successful and failed reuse attempts. The validity increases after successful reuse and decreases after failed reuse - at least temporarily until the experience is maintained respectively. Such a quantitative definition is particularly suitable for "new" experiences, i.e., experiences that are recorded directly in the EB when becoming known.

For already known experiences that are entered into the EB later, a purely quantitative definition of validity obviously cannot be complete because the number of applications before the recording in the EB cannot be determined. Instead, a qualitative definition of validity is appropriate.

Based on the ideas made explicit in the knowledge life-cycle model, the maintenance can be defined in more detail.

3.2 Deriving Maintenance Policies

The objective of deriving maintenance policies is to describe –in an informal manner– when, why, and how to do maintenance on an EB system.

This is done by deriving so-called "maintenance policies". For our purposes, we extend the definition of [24]:3 *Maintenance policies* determine when, why, and how maintenance is performed for an EB system. The "why" addresses not only the reason of maintenance but also the expected benefits of the maintenance operation, which should be related to the objectives of the EB system or to the general goal of maintenance (i.e., to preserve and improve the EB's value [32, 31]).

When DISER is used, the action refers to one task or a combination of tasks from DISER. Relevant tasks are record/update/forget experience package and restructure EB as well as their respective sub-tasks.

Maintenance policies can be derived from various sources (Fig. 2): the knowledge/ experience life-cycle model; artifacts and information gained during EB buildup; and the measurement plan for the evaluation. In addition, generic, well-tested maintenance policies from CBR research and practice should be reused for general aspects (see [24] for an overview). In the following, we describe how to derive maintenance policies from these sources and illustrate this with examples from COIN or industrial projects. For reasons of space, some maintenance policies are only outlined.

3 The original definition of [24] is as follows: "Maintenance policies determine when and how a CBR system performs case base maintenance. Maintenance policies are described in terms of how they gather data relevant to maintenance, how they decide when to trigger maintenance, the types of maintenance operations available, and how selected maintenance operations are executed."

Deriving maintenance policies from the knowledge life-cycle model

There are two major issues in the knowledge life-cycle model that are refined using maintenance policies:

1.) The transformation of experience of one type into another is performed under specific conditions and for certain reasons. These conditions and reasons have to be identified and related actions are outlined. Together reason and related action form a maintenance policy. Fig. 4 shows an example of such a maintenance policy for transforming partially tested knowledge such as lessons learned into best practice.

Trigger:	Number of lessons learned attached to a best practice description is more than X
Actions:	Aggregate best practice description with (some of the) lessons learned.
Expected benefits:	The description is more comprehensive and easier to understand.

Figure 4 Example maintenance policy for transforming experience from one type into another.

1.) The ways of dealing with failed reuse attempts have to be defined. This is defined by a combination of monitoring the quality/validity of the knowledge in the EB and proposing respective actions. These maintenance policies mainly refer to corrective actions and fix problems that were encountered during reuse of an experience.
For example, an experience was misunderstood and applied incorrectly several times, which requires two actions: rephrasing and checking if the recording is unreliable or inaccurate (see Fig. 5).

Trigger:	validity ratio < X% and number of related observations "phrasing not comprehensible or misunderstood" > N
Actions:	(a) The experience package has to be rephrased and tested regarding its understandability. The observations that are considered during rephrasing are deleted. The latter implies an increase of the validity ratio of the experience package. (b) The quality criteria for recording the knowledge have to be checked regarding their effectiveness for ensuring the comprehensibility of the recorded experiences.
Expected benefits:	The description is easier to comprehend.

Figure 5 Example of a maintenance policy derived from the knowledge life-cycle model.

For another example, the application context was too different from the contexts where the experience was gained and applied so far. Then the experience has to be split, i.e., rewritten and newly recorded for the context where it failed.

Although adding negative observations about the application of an experience decreases the validity ratio of the applied experience, this does not decrease the overall competence of the experience base in every case: The knowledge about a context where an experience is not applicable is useful to avoid making the same mistake twice and, thus, contributes to the EB's overall competence and value.

The content-triggered maintenance policies (e.g., Fig. 4 and Fig. 5) benefit in particular from the learning about maintenance. After an initial good guess for the trigger regarding X%, N, etc., these policies can be improved and validated based on the evaluation [32].

Deriving maintenance policies from conceptual model and scenarios

From the scenarios and the conceptual model, maintenance policies are derived that analyse if the different types of experience are used as intended and not "abused." Abuse is mainly possible for experience types or subject areas where users can enter items that are published without further reviewing by the EF staff.

For example: The EF allows every user to add comments to experience packages in order to make the EB more interactive. These comments should only deal with minor problems such as typos, misunderstandings, etc. A respective maintenance policy is as follows: A larger average number of larger comments indicates that the commenting feature might be abused to store actual experience without further reviews. Thus, the comments are checked. If abuse has happened, the respective comments are rewritten as experience packages and the commenting feature is deactivated. Furthermore, a more "in-time" recording of experiences could be considered because such an abuse can demonstrate the need for such a change.

Deriving maintenance policies from evaluation, user feedback, and scenarios

Maintenance policies can be stated for the different types of user feedback [9]. For example, if users do not select any experience in the query results as useful in more then X% of the queries, then the coverage could be too low or the EB might focus on the wrong subject area. Thus, the users should be interviewed respectively and the respective measure should be taken (i.e., record more experiences or change focus of EB).

For each of the reuse scenarios, the expected usage is estimated by the number of times the scenario will happen. This serves as a baseline for an evaluation of the usage of the EB.

Using strategies from CBR research and practice

Besides the tailored and more EB-specific maintenance policies that were addressed so far, maintenance strategies from CBR research and practice should be considered. Before using these, their application constraints must be checked and analysed carefully. For example, a competence-preserving approach to case deletion has been tested for planning tasks [26]. This could we reused for EB/CBR systems with a similar task.

The impact of EB objectives on the maintenance policies

The "expected benefits" section of the maintenance policies should be related to the EB's objectives or the general goal of maintenance. Since these objectives are typically very high-level, it is not very meaningful to address the EB objectives directly. Instead, we use a refinement of the objectives: the quality criteria from the evaluation program or the recording methods (e.g., see Fig. 4 and Fig. 5).

3.3 Formalizing Maintenance Policies

The maintenance policies are formalized as *maintenance guidelines* [32, 31]. Because maintenance guidelines still describe a typical task or pattern of maintenance activities, they usually refer to classes/groups of experience packages and not directly to experience packages. These maintenance guidelines are used for generating change requests for the experience packages that require maintenance. For this purpose, the following changes and extensions are made with respect to maintenance policies:

For the automatic generation of change requests, a partial or complete formalization of the "trigger" is required to allow an automatic tool-based checking of the trigger. The formalized parts of the trigger can refer to EB's conceptual model and contents as well as evaluation results. The part of the trigger than cannot be formalized is included in the maintenance guideline for manual checks by the responsible role. In case the actual trigger cannot be formalized at all, then the respective guideline can be triggered periodically as a reminder and the actual condition check is done manually by an EF staff member.

The "actions" now refer to the existing descriptions of "standard" maintenance activities as modules (if the required description already exists). EMSIG provides these modules with its maintenance primitive component. Simple text is used as glue among the modules or for the remaining parts that are not "standard."

The "expected benefits" help justify the instantiation of the guideline as change request (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 change request. In addition, the related record and reuse scenarios are stated to support the estimation of the expected benefits.

The responsible role is stated, to increase flexibility instead of a fixed assignment of experience package changes to the experience engineer and changes of the conceptual model to the experience manager. If different roles are responsible for a maintenance policy, this has to be split into several dependent maintenance guidelines to allow the assignment of the task to several persons.

Since maintenance guidelines usually refer to generic items from the EB (e.g., "process descriptions") or to a generic configuration of items. Therefore, it is necessary to generate a separate change request for each single experience package or configuration that is affected.

3.4 Tool Support for Maintenance Policies

To allow the automatic generation of change requests, tool support is essential for the maintenance guidelines. For this purpose, the maintenance assistance component of the EB maintenance and evaluation framework EMSIG [32, 31] can be used. Fig. 6 shows an example of a formalized maintenance guideline from which a change request has been generated. The reader is referred to [32] for details on the EMSIG framework and its components.

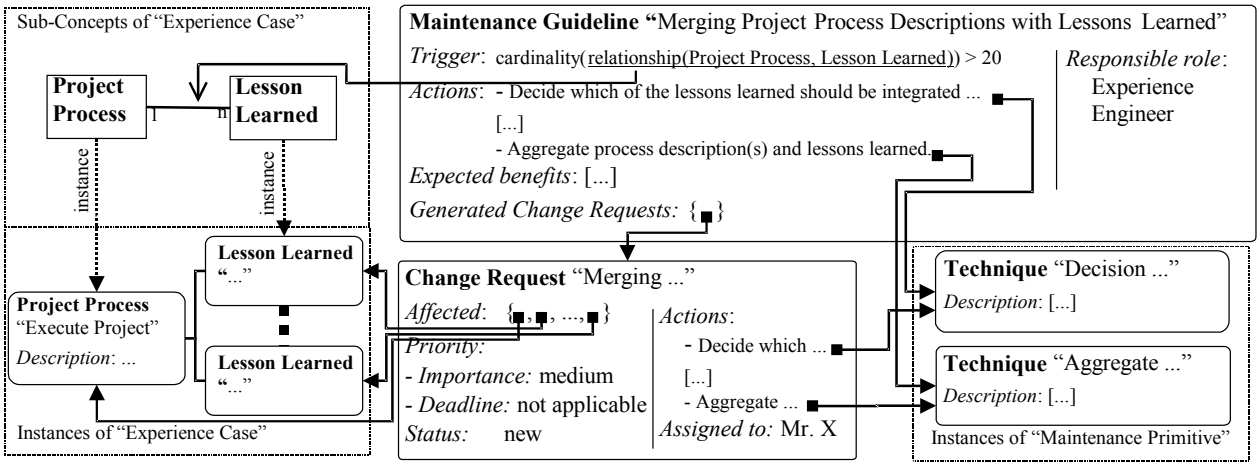

Figure 6 Example maintenance guideline (for the maintenance policy from Fig. 4) with generated change request - represented in the integrated conceptual model (Section 4.2).

4 Integrating Evaluation and Maintenance for the EB System

The maintenance and quality knowledge engineering leads to additional conceptual knowledge and cases that have to be harmonized and integrated with the conceptual model, the knowledge collection plan, and the evaluation program before the EB system is implemented.

Section 4.1 describes the integration of the conceptual model, knowledge collection plan, measurement program, and evaluation plan. Section 4.2 describes how to integrate the conceptual maintenance and quality knowledge into a more comprehensive conceptual model. Section 4.3 describes exemplary how to embed measurement data collection for evaluation into the usage on an EB system.

4.1 Harmonize & Integrate Conceptual Knowledge, Knowledge Collection Plan, and Evaluation Program

The task of harmonizing conceptual knowledge, maintenance knowledge, and quality knowledge has the objective of integrating maintenance and quality knowledge into the EB system (i.e., into conceptual model and case base).

The integration of the knowledge collection plan and of the evaluation plan is quite simple. The knowledge collection plan defines when which artifact has to be collected how by whom [45]. Thus, each entry can be represented as a maintenance guideline: "when" and "which artifact" describe the condition, "how to collect" the action, and "by whom" the responsible role. For example, if "project[X].end < 'today'" then "notify experience engineer about task 'record project experience' for project[X]".

The evaluation plan describes when which evaluation of the collected measurement data is performed how and by whom (e.g., when to analyse the collected data using a certain statistical method and when to hold a feedback session on the data analysis results with representatives of the users). Thus, the evaluation plan's structure is very similar to the knowledge collection plan's and it can be represented in the same way.

The measurement plan defines quality metrics as well as manual and automatic data collection procedures. These metrics and data collection procedures have to be integrated into conceptual model as well as usage and record scenarios (e.g., see Section 4.3). The measurement plan itself is not stored in the EB. There are two reasons: (1) The measurement plan might be kept in a separate

evaluation tool (e.g., a GQM tool). (2) The measurement plan also refers to EB aspects that are not part of the conceptual model or case base (e.g., user interface).

In the evaluation, general knowledge is identified about relationships between variation factors and quality factors. Such knowledge is attached as lessons learned to the respective parts of the record process/method description. Thus, the experience engineers are informed about these lessons when they perform the respective recording.

4.2 Integrating Maintenance and Quality Knowledge into the Conceptual Model

The *integrated conceptual model* (Fig. 7) integrates experience packages with maintenance and quality knowledge at the conceptual level. The integrated conceptual model has been implemented for COIN using the commercial CBR tool CBR-Works from empolis/tec:inno GmbH (The transition to Orenge, the successor of CBR-Works, is planned for the first half of 2001).

For each of the different types of maintenance and quality knowledge, a new top- level case concept is added. With CBR-Works, all top-level case concepts are sub-concepts of a root concept "Case." Fig. 7 depicts the types of cases at the top level, the relationships among the new concepts, and the relationships to the experience package concepts (i.e., conceptual model of actual experiences) and their respective instances (i.e., actual experience packages). The relationships reflect the structure of the maintenance guidelines and change requests from the EMSIG framework [32, 31]. 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.

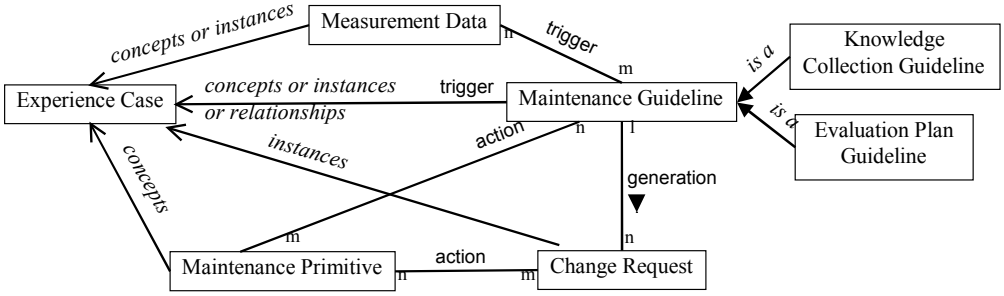

Figure 7 Framework for an integrated conceptual model of an EB including maintenance and quality knowledge (extended version of the integrated conceptual model from [32]).

The top-level case concepts can be refined and inherited as necessary. For example, all the concepts from Fig. 1 ("business process", "guideline", "prob-

lem", "observation", "project", "customer", etc.) are sub-concepts of "experience package." 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 results" (e.g., textual feedback on the whole query result and perceived usefulness of a retrieved case [33]). The concept "maintenance guideline" has "knowledge collection guideline" as sub-concept. These are all the maintenance guidelines that form the knowledge collection plan. The EF staff can retrieve the knowledge collection plan from the case base using a simple retrieval on all cases of the "knowledge collection guideline" concept. Fig. 6 shows -by examples- how maintenance knowledge is represented using the integrated conceptual model.

4.3 Embedding Evaluation into Usage

Practice has shown that the users' motivation for entering measurement data during usage is rather low. A reason for that is certainly that it is not practically possible to involve all users in the development/definition of the measurement program.

This means that (a) if possible, measurement data should be collected automatically (i.e., without further user interaction), and (b) the collection of measurement data that can only be collected manually has to be combined with useful add-on features according to the principle "we want measurement data from the user, we offer him something."

An opportunity to smartly combine the collection of measurement data as feedback with the usage process is to establish a so-called "feedback loop" (Fig. 8). In the presented example, the collection of feedback is integrated into the project process in a simple manner. It is part of the project planning, in the beginning of the project, to identify the existing, relevant experiences in COIN. This is done using similarity-based retrieval over IESE's intranet. The project manager receives as an answer to his query a list of 30 similar experiences, which he can classify as useful or not useful. With a click, he composes a checklist of these useful experience for his project. This checklist can be printed or emailed.

While this classification collects feedback on the estimated/expected usefulness before the application of the experiences, the project analysis interview is used for asking about the actual usefulness of the respective experiences. The analysis of usage and usefulness of the experience packages delivers information that is used for (1) empirically validating the experience packages and (2) maintaining the EB.

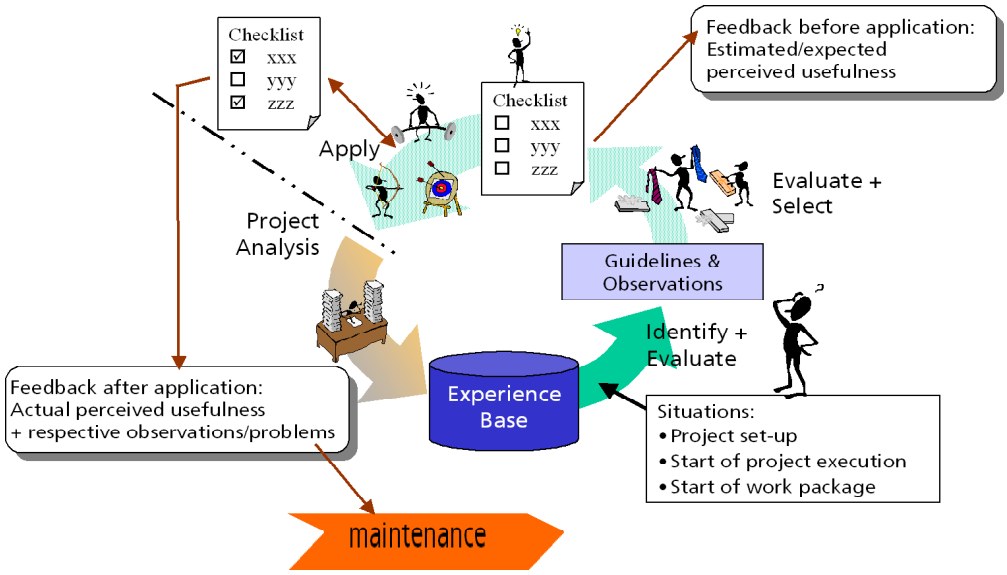

Figure 8 Embedding the collection of measurement data in the usage process (example).

5 Discussion: Engineering Maintenance and Quality Knowledge during EB Buildup

This section discusses how to integrate the method into an EB buildup methodology (namely DISER [45] and INCREA [15]). This allows to systematically develop the required maintenance and quality knowledge during EB buildup. On the one hand, this ensures that the maintenance and quality knowledge is acquired as early as possible and that the maintenance and quality needs are considered during the development of the EB in order to avoid future changes due to maintenance needs identified later than possible. On the other hand, we observed that the conceptualization usually is not stable during buildup and prototypical usage [33]. Thus, evaluation and maintenance should be addressed with low-effort solutions, i.e., generic or standard components should be used for maintenance and evaluation where possible and feasible.

Since the development of the EB itself is iterative and does not follow a waterfall approach, it is not very useful to link the definition and elicitation of maintenance and quality knowledge too strictly to the EB development process. In addition, mixing everything can lead to a "cognitive overload" for the experts involved because they have to make too many decisions with different background (experience itself, maintenance, evaluation/quality). Therefore, the development information and artifacts are grouped into three levels:4 high-level information dealing with the objectives, success criteria, and subject areas; mid-level information dealing with detailed scenarios, conceptual model, and record methods; and low-level information dealing with the technical implementation including the design of the user interface.

The knowledge life-cycle model is considered as high-level information. In particular, it should be discussed and defined after the subject areas and before the detailed scenarios because the life-cycle model defines together with the subject areas –at an abstract level– which types of knowledge and experience are in the focus of the EB and, thus, have to be addressed by the detailed scenarios.

The maintenance policies (related to the knowledge life-cycle model or to the scenarios and conceptual model) are conceptual information. Obviously, they require that the conceptual model of the actual experience is settled. Therefore, they should be derived after the definition of the conceptual model for the actual experience.

4 The terminology is taken from DISER. However, the terms in INRECA are almost the same.

The formalization of the maintenance policies into maintenance guidelines and the automation is part of the technical implementation of the EB system.

A feedback loop -as presented in Section 4.3- obviously has an impact on the design of the EB's user interface and on the knowledge collection. Thus, a decision on a feedback loop should be made before addressing the latter two issues.

The measurement plan must be completed before doing the harmonizing task from Section 4.1 because it is input for this task. For the reasons mentioned above, a generic standard measurement plan should be used. Such a measurement program was outlined in [32]. It uses two indicators for the EB value: sustained usage of the EB and perceived usefulness of the retrieved cases. Together with the feedback loop from Section 4.3, it has been implemented for COIN.

A measurement program also includes the definition of a baseline or an estimation of the quality criteria under focus. For the usage, the expected number can be determined after defining the scenarios. This also helps to decide on scenarios that are not relevant due to a low number of expected uses.

6 Conclusion

In [32, 31], we presented an evaluation and maintenance methodology for experience bases (EBs). This methodology is based on two ideas: (1) EB maintenance is driven by systematically conducting and exploiting evaluation. (2) EB maintenance itself is performed systematically by recording and using experience gained during maintenance in the form of special maintenance guidelines.

This paper extends this maintenance and evaluation methodology with a method for the systematic development of maintenance and evaluation knowledge during EB buildup. Operational maintenance knowledge is systematically derived from various sources such as artifacts and information gained during EB buildup, a knowledge/ experience life-cycle model, and the evaluation program for the EB. In addition, generic, well-tested maintenance policies from CBR research and practice should be reused for general aspects (see [24] for an overview). The maintenance knowledge is formalized further to enable automated support. For the implementation, the maintenance and quality knowledge is integrated into the EB. The collection of quality/measurement data is embedded into the usage to increase the motivation of the users for providing quality/ measurement data. The systematic development has the benefit that the maintenance knowledge can be traced to its roots and that a detailed operational maintenance plan/ strategy is developed. This explicit relation of the maintenance policies and guidelines to the EF/EB's objectives and quality criteria ensures that the maintenance knowledge addresses relevant issues of maintenance. The fine-tuning of the maintenance knowledge for an EB is an issue of learning about EB maintenance [32].

We used the presented method for systematically developing maintenance knowledge for COIN and an industrial EF project in the telecommunication section. Here we could also transfer maintenance knowledge (in form of maintenance policies) from one EB to another. This transfer and reuse was simplified by the similarity of knowledge life-cycle model and conceptual model (at a coarse-grained level).

We also used the GQM method for developing quality knowledge and transferring this quality knowledge across EFs. We used our experience gained in [33, 34] to set up evaluation programs for COIN and industrial projects faster and cheaper. The standard set of metrics allows better comparison of evaluation results across EFs.

As a next step, the proposed integration of the maintenance and quality engineering methods into the buildup method DISER will be tested. Furthermore, future work will deal with a systematic analysis of maintenance reasons and respective actions in a generic knowledge life-cycle model for all kinds of knowledge and experience (i.e., set of generic maintenance policies related to the life-cycle model) considering existing maintenance knowledge and its dimensions [39, 24, 44]. A collection of such knowledge will jump-start the definition of maintenance policies related to the knowledge life-cycle model of EBs.

Acknowledgements

The COIN project has been funded as an internal project at IESE since January 2000.

References

[1] Aamodt, A. Towards robust expert systems that learn from experience - an architectural framework. In Boose, J., Gaines, B., and Ganascia, J.-G., editors, *Proceedings of the Third European Workshop on Knowledge Acquisition for Knowledge Based Systems*, pages 311– 326, 1989.

[2] Abecker, A., Decker, S., and Kühn, O. Organizational memory (in German). *Informatik-Spektrum*, 21(4):213–214, Aug. 1998.

[3] Aha, D., and Weber, R., editors. Proceedings of the Workshop on Intelligent Lessons Learned Systems at 17th Na¦tional Conference on AI (AAAI-00), 2000.

[4] Aha, D. W. The AAAI-99 KM/CBR workshop: Summary of contributions. In *Proceedings of the ICCBR 99 Workshops*, pages II–37–II–44. Department of Computer Sci¦ence, University of Kaiserslautern: Centre for Learning Systems and Applications, 1999. Technical Report, LSA-99-03E.

[5] Althoff, K.-D. Case-based reasoning. In Chang, S. K., editor, *Handbook of Software Engineering and Knowledge Engineering*, volume 1. World Scientific, 2001. (to appear).

[6] Althoff, K.-D., Birk, A., Hartkopf, S., Müller, W., Nick, M., Surmann, D., and Tautz, C. Systematic population, utilization, and maintenance of a repository for comprehensive reuse. In Ruhe, G., and Bomarius, F., editors, *Learning Software Organizations - Methodology and Applications*, number 1756 in Lecture Notes in Computer Science, pages 25–50. Springer Verlag, Heidelberg, Germany, 2000.

[7] Althoff, K.-D., Bomarius, F., and Tautz, C. Using case-based reasoning technology to build learning organizations. In *Proceedings of the the Workshop on Organizational Memories at the European Conference on Artificial Intelligence '98*, Brighton, England, Aug. 1998.

[8] Althoff, K.-D., Kockskämper, S., Maurer, F., Stadler, M., and Wess, S. Ein system zur fallbasierten wissensverarbeitung in technischen diagnosesituationen. In Retti, J., and Leidlmeier, K., editors, *5th Austrian Artificial-Intelligence-Conference*, pages 65–70. Springer Verlag, 1989.

[9] Althoff, K.-D., Nick, M., and Tautz, C. Improving organizational memories through user feedback. In *Workshop on Learning Software Organisations at SEKE'99*, Kaiserslautern, Germany, June 1999.

[10] Althoff, K.-D., and Wess, S. Case-based reasoning and expert system development. In Schmalhofer, F., Strube, G., and Wetter, T., editors, *Contemporary Knowledge Engineering and Cognition*, pages 146–158. Springer Verlag, 1992.

[11] Bartsch-Spörl, B. Ansätze zur Behandlung von fallorientiertem Erfahrungswissen in Expertensystemen. *Künstliche Intelligenz*, 4:32–36, 1987.

[12] Basili, V. R., Caldiera, G., and Rombach, H. D. Experience Factory. In Marciniak, J. J., editor, *Encyclopedia of Software Engineering*, volume 1, pages 469–476. John Wiley & Sons, 1994.

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

[14] Bergmann, R. Experience management - foundations, development methodology, and internet-based applications. Postdoctoral thesis (submitted), Department of Computer Science, University of Kaiserslautern, 2001.

[15] Bergmann, R., Breen, S., Göker, M., Manago, M., and Wess, S. *Developing Industrial Case-Based Reasoning Applications – The INRECA Methodology*. Springer Verlag, 1999.

[16] Birk, A., and Tautz, C. Knowledge management of software engineering lessons learned. In *Proceedings of the Tenth Conference on Software Engineering and Knowledge Engineering*, San Francisco Bay, CA, USA, June 1998. Knowledge Systems Institute, Skokie, Illinois, USA.

[17] Buglione, L., and Abran, A. Balanced scorecards and GQM: What are the differences? In *Proceedings of the Third European Software Measurement Conference (FESMA-AEMES 2000)*, Madrid, Spain, Oct. 2000.

[18] Davenport, T. H., and Prusak, L. *Working Knowledge. How Organizations Manage What They Know*. Harvard Business School Press, Boston, USA, 1998.

[19] Goodall, A. Survey of knowledge management tools - part i & ii. *Intelligence in Industry*, 8, January/February 1999.

[20] Hammer, M., and Champy, J. *Reengineering the Corporation*. Nicolas Brealey Publishing, London, 1993.

[21] Johansson, C., Hall, P., and Coquard, M. "talk to paula and peter - they are experienced" - the experience engine in a nutshell. In Ruhe, G., and Bomarius, F., editors, *Learning Software Organi¦zations - Methodology and Applications*, number 1756 in Lecture Notes in Computer Science, pages 171–185. Springer Verlag, 2000.

[22] Kaplan, R. S., and Norton, D. P. *The Balanced Scorecard. Translating Strategy into Action*. Harvard Business School Press, Boston, 1996.

[23] Leake, D. B., Smyth, B., Wilson, D. C., and Yang, Q., editors. *Computational Intelligence special is¦sue on maintaining CBR systems*, 2001. (to appear).

[24] Leake, D. B., and Wilson, D. C. Categorizing case-base maintenance: Dimensions and directions. In Smyth, B., and Cunningham, P., editors, *Advances in Case-Based Reasoning:Proceedings of the Fourth European Workshop on Case-Based Reasoning*, pages 196–207, Berlin, Germany, Sept. 1998. Springer-Verlag.

[25] Lehner, F. Organisational Memory - Konzepte und Systeme für das organisatorische Lernen und das Wissensmanagement. Carl Hanser Verlag, 2000.

[26] McKenna, E., and Smyth, B. Competence-guided editing methods for lazy learning. In Horn, W., editor, *Proceedings of the 14th European Conference on Artificial Intelligence*. IOS Press, Berlin, Germany, 2000.

[27] Menzies, T. "knowledge maintenance: The state of the art". "*The Knowledge Engineering Review*", 1998.

[28] Minor, M., Funk, P., Roth-Berghofer, T., and Wilson, D., editors. Proceedings of the Workshop on Flexible Strategies for Maintaining Knowledge Containers at the 14th European Conference on Artificial Intelligence (ECAI 2000), Aug. 2000.

[29] Minor, M., and Hanft, A. Corporate knowledge editing with a life cycle model. In *Proceedings of the Eighth German Workshop on Case-Based Reasoning*, Laemmerbuckel, Germany, 2000.

[30] Müller, M. Interestingness during the discovery of knowledge in databases (in German). *Künstliche Intelligenz*, pages 40–42, Sept. 1999.

[31] Nick, M., and Althoff, K.-D. Systematic evaluation and maintenance of experience bases. In Minor, M., editor, *ECAI Workshop Notes: Flexible Strategies for Maintaining Knowledge Containers*, Berlin, Germany, 2000.

[32] Nick, M., Althoff, K.-D., and Tautz, C. Systematic maintenance of corporate experience repositories. *Computational Intelligence special issue on maintaining CBR systems*, 2000. (accepted).

[33] Nick, M., and Feldmann, R. Guidelines for evaluation and improvement of reuse and experience repository systems through measurement programs. In *FESMA-AEMES 2000*, Madrid, Spain, Oct. 2000.

[34] Nick, M., and Tautz, C. Practical evaluation of an organizational memory using the goal-question-metric technique. In *XPS'99: Knowledge-Based Systems - Survey and Future Directions*. Springer Verlag, Würzburg, Germany, Mar. 1999. LNAI Nr. 1570.

[35] Nonaka, I. The knowledge-creating company. *Harvard Business Review*, 69(6):96–104, No¦vember/December 1991.

[36] Plaza, E., and de Mantaras, R. L. A case-based apprentice that learns from fuzzy exam¦ples. In Ras, Zemankova, E., editor, *Methodologies for Intelligent Systems*, volume 5, pages 420–427. North Holland, 1990.

[37] Reinartz, T., Iglezakis, I., and Roth-Berghofer, T. On quality measures for case base maintenance. In Blanzieri, E., and Portinale, L., editors, *Advances in Case-Based Reasoning:Proceedings of the Fifth European Workshop on Case-Based Reasoning*, pages 247–259. Springer-Verlag, 2000.

[38] Richter, M. M. Introduction. In Lenz, M., Bartsch-Spörl, B., Burkhard, H.-D., and Wess, S., editors, *Case-Based Reasoning Technologies: From Foundations to Applications*, number 1400 in Lecture Notes in Artificial Intelligence, chapter 1, pages 1–15. Springer- Verlag, Berlin, Germany, 1998.

[39] Rombach, H. D. Keynote: Maintenance of software development/maintenance know-how. In *Proceedings of the International Conference on Software Maintenance*, 1995.

[40] Romhardt, K. Die Organisation aus der Wissensperspektive - Möglichkeiten und Grenzen der Intervention. Gabler Verlag, Wiesbaden, 1998.

[41] Ruhe, G. Learning software organisations. In Chang, S. K., editor, *Handbook of Software Engineering and Knowledge Engineering*, volume 1. World Scientific, 2001. (to appear).

[42] Sary, C. Recall prototype lessons learned writing guide. Technical Report 504-SET-95/003, NASA Goddard Space Flight Center, Greenbelt, Maryland, USA, Dec. 1995.

[43] Schank, R. C. Dynamic Memory: A Theory of Learning in Computers and People. Cambridge University Press, 1982.

[44] Swanson, E. B. The dimensions of maintenance. In *Proceedings of the Second International Conference on Software Engineering*, pages 492–497, 1976.

[45] Tautz, C. Customizing Software Engineering Experience Management Systems to Organizational Needs. PhD thesis, University of Kaiserslautern, Germany, 2000.

[46] Tautz, C., Althoff, K.-D., and Nick, M. A case-based reasoning approach for managing qualitative experience. In *AAAI-00 Workshop on Intelligent Lessons Learned Systems*, 2000.

[47] van Heijst, G., van der Speck, R., and Kruizinga, E. Organizing corporate memories. In Proceedings of the Tenth Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, 1996. URL http://ksi.cpsc.ucalgary.ca/KAW/KAW96/vanheijst/HTMLDOC.html.

Document Information

Title:	Engineering Experience Base Maintenance Knowledge
Date:	March 17, 2001
Report:	IESE-01.01/E
Status:	Final
Distribution:	Public