Fraunhofer Einrichtung
Experimentelles
Software Engineering

Improving Organizational Memories Through User Feedback

Authors:
Klaus-Dieter Althoff
Markus Nick
Carsten Tautz

Accepted for publication in
Proceedings of the Workshop on
Learning Software Organizations,
Kaiserslautern, June 1999

IESE-Report No. 004.99/E
Version 1.1
February 1, 1999

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

Executive Summary

The benefits of an organizational memory are ultimately determined by the usefulness of the organizational memory as perceived by its users. Therefore, an improvement of an organizational memory should be measured in the added perceived usefulness. Unfortunately, the perceived usefulness has many impact factors (e.g., the precision of the user query, the urgency with which the user needs information, the coverage of the underlying knowledge base, the quality of the schema used to store knowledge, and the quality of the implementation). Hence, it is difficult to identify good starting points for improvement.

This paper presents the goal-oriented method OMI (Organizational Memory Improvement) for improving an organizational memory incrementally from the user's point of view. It has been developed through several case studies and consists of a general usage model, a set of indicators for improvement potential, and a cause-effect model. At each step of the general usage model of OMI, protocol cases are recorded to pinpoint improvement potential for increasing the perceived usefulness. If improvement potential is identified based on the interpretation of the protocol cases, the user is asked for specific improvement suggestions. Since this is a continuous diagnosis of the usefulness of the organizational memory, this method is able to adapt the organizational memory to the needs of the users – even if the environment, for which the organizational memory was designed, changes.

Table of Contents

1 Introduction

The management of knowledge is a critical factor of an enterprise´s success. The objective of knowledge management is the optimal use of the resource »knowledge« for enabling learning from experience, continuous process improvement, and the extension of a company's creativity potential [ADK98]. To support these objectives, a company's knowledge has to be explicitly stored in a so-called organizational memory (OM). While building up such an OM is already a challenging task and involves difficult subtasks like knowledge acquisition, modeling, evaluation, and reuse, the improvement of an OM through user feedback is of essential importance if an OM shall be a continuous source of a company's benefit.

The benefits of an OM are ultimately determined by the usefulness of the OM as perceived by its users. Therefore, an improvement of an OM should be measured in the added perceived usefulness. Unfortunately, the perceived usefulness has many impact factors (e.g., the precision of the user query, the urgency with which the user needs information, the coverage of the underlying knowledge base, the quality of the schema used to store knowledge, and the quality of the implementation). Hence, it is difficult to identify good starting points for improvement.

This paper presents the goal-oriented method OMI (Organizational Memory Improvement) for improving an OM incrementally from the user's point of view. It has been developed through several case studies [ABT98] and consists of a general usage model, a set of indicators for improvement potential, and a cause-effect model. At each step of the general usage model of OMI, data in the form of protocol cases are recorded to pinpoint improvement potential for increasing the perceived usefulness. Such protocol cases include formal parts like the user-specified query, the retrieval goal of the query (see Section 3), the result of the query, and certain actions the OMI method has invoked (e.g, asking the user a specific question or forwarding the case to the OM maintenance team). The protocol cases also include informal parts like the user´s answers to certain questions, the user´s general feedback at the end of a session, or some explanatory texts that have been written by the OM maintenance team. The purpose of protocol cases is twofold:

- On the one hand they can be used by the OM maintenance team to improve the OM's content, to react on identified user problems by contacting the respective user, or to start any necessary improvement action.
- On the other hand, if a protocol case – or parts of it – have been qualified and published by the OM maintenance team, it can be used by the OM man-

agement system (OMMS). Based on the general usage model of OMI and the formal parts of the respective protocol case, the OMMS can automatically interpret the case, i.e., apply case-based reasoning (CBR). This includes asking the user specific questions to collect his feedback or presenting situation-specific lessons learned.

If improvement potential is identified (by the OM maintenance team or the OMMS) based on the interpretation of the protocol cases, the user is asked for specific improvement suggestions. Since this is a continuous diagnosis of the usefulness of the OM, this method is able to adapt the OM to the needs of the users – even if the environment, for which the OM was designed, changes. Thereby the OMI method allows to overcome the general problem that users are often not available during the development of the OM. The basic idea underlying OMI is to start with an »intelligent guess« (not focus of this paper) and to improve the OM/OMMS incrementally by systematically collecting and analyzing focused user feedback (focus of this paper). Basing on the OL-CBR approach (organizational level case-based reasoning: [ABT98]), OMI integrates the manual interpretation of the protocol cases by the OM maintenance team with the automatic interpretation by the OMMS, which actually is based on a CBR tool (see Section 2). The more formal knowledge is recorded as part of the protocol cases (based on OMI's usage model) and/or added through the OM maintenance team´s analysis and revision, the more knowledge can be used by the OMMS to adapt to the users´ needs based on user feedback. Thereby OMI can cope with practical constraints like avoiding user overload through asking too many questions.

In spirit OMI is comparable to evolutionary constructions of repositories (e.g., [Hen97]) and approaches based on Failure Mode and Effect Analysis (FMEA) [JK98, PZ96, Pfe96], which also try to avoid failure situations, i.e. in case of OMI, a decrease of the perceived usefulness. At least partially these approaches base also on CBR [JK98, PZ96].

OMI can also be compared to approaches for diagnostic problem solving, especially case-based diagnosis [AW91, Wes95, Alt97], because a classification problem has to be solved based on collected symptoms. By contrast, OMI is not intended to be fully automated. It is flexible concerning the degree of using fully automated interpretation of cases by the OMMS or manual case analysis and interpretation by the OM maintenance team.

In the next section our approach for implementing an OM is shortly introduced. Section 3 and Section 4 present OMI's cause-effect model for usefulness as perceived by the user and the general usage model, respectively. Section 5 describes OMI's diagnostic process for improving an OM, whereas Section 6 explains the utilization of protocol cases. Finally some conclusions are drawn.

2 Our Implementation: A Case-Based Organizational Memory

To talk about improvement of an OM, we need to state some assumptions about implementation aspects of the OM. We will do so by describing major primitives of the representation formalism REFSENO[1] we use for specifying the ontology (i.e., the schema) underlying an OM [ABT98, TG98, TG99]. REFSENO is implemented in collaboration with a commercial CBR tool provider [Tec99][2].

1. An OM contains different types of experience (e.g., lessons learned, typical effort distribution for projects, specific project information such as schedules and milestone plans). Each of these types is specified by a *concept*. The actual contents of the OM are stored as instances of the concepts. Hence, a concept describes a class of experience items. Figure 1 describes the three-layered representation we employ. The meta knowledge (in the form of REFSENO) guides the specification of an ontology for an OM, whereas the ontology guides the population and exploitation of the OM's contents.

2. The OM's contents are made up of two parts: the artifact itself and its characterization. The artifact itself can be a file or database. Often it has a propriety format, meaning that it is difficult or impossible to access the information contained in it by generic tools (like an OMMS). The characterization part contains all information that is deemed to be necessary for performing some predefined task(s) with the experience item. For instance, a lesson learned may be characterized for the purpose of later application by information about the lesson learned itself (e.g., name, length, descriptions of a problem and one or more solutions for solving the problem), information about its interface (e.g., prerequisites for the application of the lesson learned), and information about the context in which it has been created and used (e.g., project and situation in which the described problem occurred). These characterizations guide both storing and searching for experience items. Information contained in characterizations may overlap with or complement information contained in the artifact [ABT98].

3. It is *partially* known a-priori what information needs to be part of the characterization.[3] This knowledge is captured as part of the ontology in the form of a set of *attributes*. We distinguish two categories of attributes: *terminal* (containing actual information needed for some task) and *nonterminal attributes* (acting as references to related instances which in turn contain more terminal

1 REFSENO stands for *Re*presentation *F*ormalism for *S*oftware *En*gineering *O*ntologies.
2 A first prototype implementation can be accessed at http://demolab.iese.fhg.de:8080/.
3 During operation, further information needs will become apparent. Considering them constitutes an improvement of the OM.

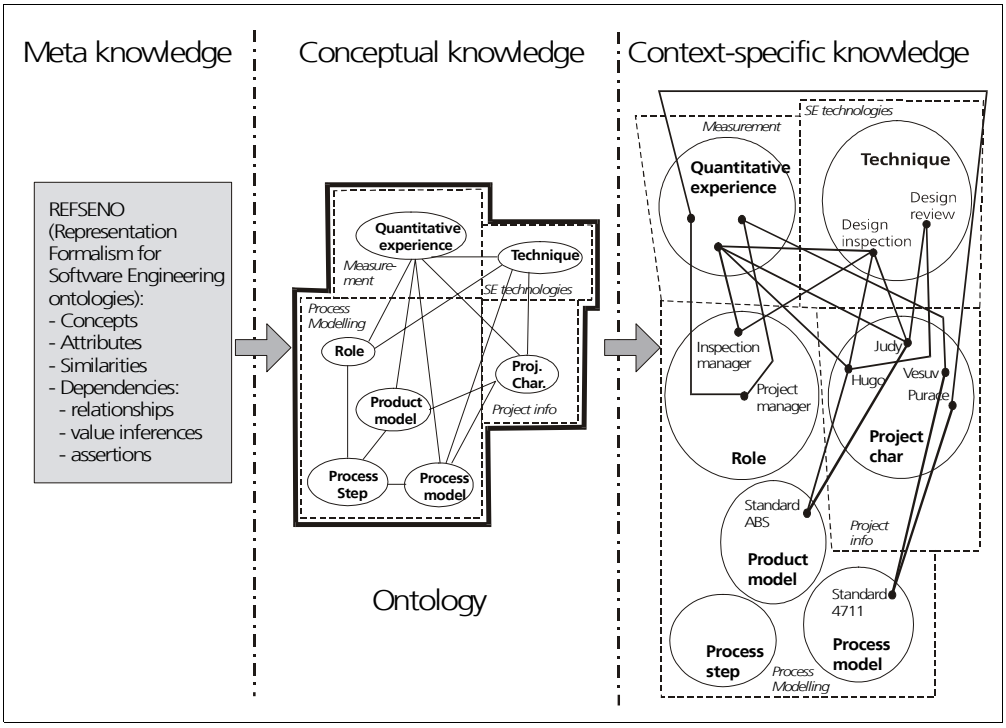

Figure 1: Three types of knowledge are used to describe schema and contents of an organizational memory.

and nonterminal attributes; represented by lines in Figure 1). Terminal attributes can be used for retrieval, whereas nonterminal attributes can be used for navigational search. Just as in modern programming languages, both categories of attributes can be typed. This gives the OMMS the possibility to perform certain consistency checks on the contents of the OM (e.g., whether a value supplied by the user lies within a predefined value range or whether the value of a nonterminal attribute references an instance of a predefined concept). They also allow to perform type checks for value inferences (formulas that specify how to calculate automatically computable attribute values) and assertions (formulas that specify dependencies between a set of attributes).

4. Exact matches between a query and the instances in the OM can be expected only in rare situations. This leads to the introduction of similarity. If an OMMS supports retrieval based on characterizations, it must be capable of returning characterizations similar to the one used for specifying the needed artifact. The underlying hypothesis is that two artifacts (e.g., milestone plans) are similar if they have similar characterizations. Hence, this assumption allows us to retain experience items as they are gained, that is, as instances of predefined concepts, keeping the effort for storing new experience low. During reuse, stored experience can be retrieved (via the similarity feature) and adapted to

the new situation. In the ontology, similarity functions are associated with attributes (local similarity) and concepts (global similarity combining local similarities).

In summary, the structural specification of an OM is made up of a set of concepts that are specified through a set of typed attributes:

$$\text{Spec}_{\text{OM}} = \{\{c_1, \ldots, c_n\}, \{t_1, \ldots, t_l\}\}$$

$$\text{with} \quad c_j = \left\{a_{j,1}, \ldots, a_{j,m_j}\right\} (j \in \{1, \ldots, n\})$$

$$\text{where} \quad \text{type}(a_{j,k}) \in \{t_1, \ldots, t_l\} (k \in \{1, \ldots, m_j\})$$

There is a similarity function associated with the concept and the attributes. The similarity is always computed from an instance i to a query q where i and q are instances of the same concept, i.e.:

$$\text{concept}(i) = \text{concept}(q) = c = \{a_1, \ldots, a_m\} \in \{c_1, \ldots, c_n\}$$

It is computed using the similarity function associated with this concept:

$$\text{sim}(i, q) = \text{sim}_c(i, q) = f\left(\text{sim}_{a_1}\left(\text{val}_{a_1}(i), \text{val}_{a_1}(q)\right), \ldots, \text{sim}_{a_m}\left(\text{val}_{a_m}(i), \text{val}_{a_m}(q)\right)\right)$$

where $\text{val}_a(i)$ stands for the value of attribute a of instance (or query) i.

3 Perceived Usefulness

The success of an OM can be measured in many ways. Examples for specific views on evaluation mainly from the knowledge-based system and related fields are [AA96, Alt97, Coh89, GKP+83, GXG98, Kir94, SW88, vW96]. Also, some evaluation work has been done in the area of software reuse (programs), mainly regarding economic success [BB91, Lim96, HS93]. Many of the economic models for software reuse can also be used to evaluate organizational memories, because OMs are also about reuse. Only in the case of an OM, reuse is not restricted to software development artifacts. Other evaluation criteria, most importantly *recall* and *precision*, come from library and information science [SM83].

However, using a goal-oriented measurement and evaluation approach where experts participate in the definition of a measurement program [BDR96], we found out that the usefulness, as perceived by the user of the OM, is the most important measure for the success of an OMMS [NT99]. This is not surprising since an OM is worthless if it fails to deliver information that is of value to its users. These findings are also supported by Harter [Har92] (there called »psychological relevance«) and Cooper [Coo97] (there called »personal utility«).

Therefore, we will judge whether any change is an actual improvement based on the perceived usefulness *before* and *after* the change. If the usefulness improved, the change is regarded as an improvement.

As pointed out by Cooper, the ideal measurement of the usefulness as perceived by the user is practically and economically impossible [Coo97]. Therefore, we have to simplify the measurement procedure. To do so, we recall the meaning of similarity. Ideally, the similarity between an experience item in an OM and the needed experience (specified by the query) is an a-priori approximation of the a-posteriori usefulness as experienced by the user [Wes95, AR99]. If the OMMS returns the instances $i_1, \ldots, i_n$ in response to a query q, where i_1 is most similar to q and i_n least similar, the user should select (ideally) i_1 or – if more than one instance is perceived as useful – the set $i_1, \ldots, i_m$ with $m \leq n$. Ideally, $m = n$. This implies also an assignment of the degree of usefulness to the instances on an ordinal scale, that is, i_1 is the most useful instance, i_2 the second most useful, etc.

Since it is difficult to determine a minimal similarity value (this depends among others on the background knowledge of the user), an OMMS could return a fixed number of instances, e.g., $n = 10$. If i_m denotes the last *useful* instance, then the system is optimally useful if an OMMS never returns an instance $i \in \{i_1,$

…, i_m} that is not useful. The more often an OMMS returns such an instance, the less useful it will be. Using this definition, the user can simply mark all instances i_1, …, i_n as either useful or not useful. Based on this data, the usefulness of an OMMS can be computed (e.g., relative to the number of queries issued by users). Another important aspect of usefulness is that an OMMS returns useful instances at all, i.e., m should be greater than 0.

Unfortunately, there is no single parameter with which the usefulness of an OMMS can be changed. Rather, there is a large number of variation factors. Figure 2 shows the variation factors we have identified so far. However, we do not claim that the list is exhaustive. In the following, we will go into detail for the main branches of the cause-effect model.

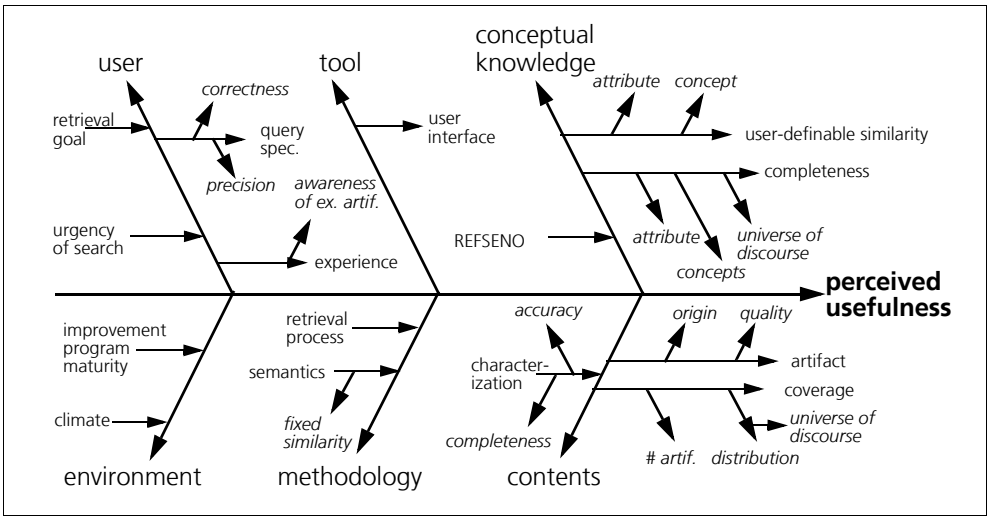

Figure 2: OMI's cause-effect model: The usefulness as perceived by the user is influenced by many factors.

- **User**. The intentions and abilities of the user will greatly influence the usefulness of the experience items returned by the OMMS. For instance, if a user does not specify his needs correctly, the OMMS might return experience items adequate for the needed experience as specified by the query, but not for the actual experience needed (which is in the mind of the user). But even if the query is specified correctly, it may be underspecified (user gives too little attribute values to allow a meaningful differentiation among the stored experience items) or overspecified (the user gives too many attribute values – no experience item in the OM can be found which matches the query). An overspecification is only possible in OMMS that are only capable of retrieving exact matches or OMMS computing a cutoff-value for similarities. For a similarity-based OMMS that returns a fixed number of instances, an overspecification is practically impossible.

  Also of importance for the usefulness of an experience item is the purpose for which the user retrieves the experience (denoted as »retrieval goal« in

Figure 2). A characterization may contain all information that is necessary to apply a lesson learned (thus the lesson learned will be perceived as useful), but may fail to provide hints how to adapt its solution to other situations (thus the lesson learned might be perceived as not useful if it needs to be changed).

Quite interestingly, the urgency of the search [Har92] and the experience of the user (e.g., the user might perceive only new experience, that is, experience he was not aware of, as useful) [Coo97] will also affect the perceived usefulness.

- **Environment**. The environment in which the OMMS operates may be mature. In this case the OM tends to be filled with more experience that has been gained by the organization itself. Typically, own experience is more valuable than experience that is part of textbooks, because it can be tailored more easily to new situations since the contexts between the situation in which it was gained and the situation in which it is applied do not differ as much (or at least, the differences are better known) [BR88, BCR94]. Also, the climate in an organization influences how willingly people are to share their experience with their colleagues over an OMMS. In organizations where mistakes are not viewed as chances to learn, valuable information will remain in the minds of the people – mistakes will be repeated [Dam98].

- **Tool**. For the user, the tool (OMMS) is mainly characterized through its behavior and its user interface. The behavior is determined through all other branches (except for the »environment« branch) and is not considered further at this point. The user interface constitutes a sort of barrier for the usage of a system. If a system is hard or cumbersome to use, people will try avoid using the system [MD97, Nic98, NT98]. And, if a system is not used, it is perceived as not useful.

- **Methodology**. It is the methodology that is supported by the OMMS. If the underlying methodology (e.g., the retrieval process supported by the OMMS or the semantics of REFSENO's primitives) is not optimal, it cannot be expected that the system is perceived as useful as it could be.

- **Conceptual Knowledge**. Clearly, the behavior of an OMMS is not solely determined by its implementation, but also by its contents and the organization of these. As the conceptual knowledge determines what and how experience is stored in the OM, it plays a major role regarding the usefulness of a system. First of all, the concepts and their attributes define a universe of discourse that the OM *can* cover. In contrast to this universe stands the universe of discourse the OM *shall* cover. The concepts and attributes may not cover all kinds of experience to be stored or all information needed to perform certain predefined tasks. In addition, the similarity functions may not approximate the perceived usefulness appropriately. And finally, certain characteris-

tics of the universe of discourse may not be expressible using the primitives of REFSENO.

- **Contents**. Even if the conceptual knowledge is defined optimally regarding the usefulness, the contents of the OM may cause the system to fail. Just as with databases, the information stored in an OM must be accurate and complete [NT98]. Otherwise, users will lose their confidence in the experience provided by the system. This will lower the overall perception of the system's usefulness.

  Also, the universe of discourse the OM shall cover is not covered simply by defining the universe in terms of experience types to be stored. The actual experience has still to be stored! The coverage of a universe of discourse is influenced by two characteristics: the number of artifacts in the OM and their distribution. The more artifacts are stored, the *more likely* it is that the system will return useful information. However, it is possible that many artifacts are stored, but the stored artifacts do not match the users' queries good enough (they are not similar enough). Thus, the distribution of the artifacts must match the distribution of the users' queries.

  The usefulness of experience offered by the OMMS is also determined by the known quality of its artifacts. Quality can be measured in many ways, e.g., in terms of popularity or importance [Coo97]. In addition, the origin of an artifact (e.g., the author) may influence the perceived usefulness. For instance, assume that the user issues a search request on software inspections. Let us further assume that the OMMS returns experience on software inspections whose author is known to the user to be an expert in software testing. The user may now value the usefulness of this particular experience item very high, because he may suspect a connection between software inspections and testing. In a consecutive query, the user may now also want to include experience regarding software testing.

4 Usage Model

As can be seen from the variation factors presented in the previous section, the usage of an OMMS cannot be described by a sequential process. The result of a query may lead to new insights and thus to additional queries. Queries leading to unsatisfactory results will be changed and issued again. To analyze these situations further, we first describe the »ideal« (i.e., sequential) usage scenario for an OMMS. Later, we describe under which circumstances the user might go back and repeat some of the steps.

Figure 3 shows the sequential usage model. The process starts with the specification the user has in mind for the experience needed. The user formulates a query based on this specification guided by the ontology. This query is input for the OMMS. It identifies potential experience items (e.g., if the user requests a project schedule, all project schedules are potential experience items). The identified experience items are then evaluated by computing a similarity value (as defined by the ontology) for each of the experience items to the query. The resulting list is ordered by decreasing similarity and cut off at a fixed number (e.g., after the tenth experience item). The characterizations of the ten artifacts are displayed to the user who evaluates (manually) the offered experience items. The user selects and retrieves all useful artifacts from the OM based on the provided characterizations. Finally he applies the artifacts.

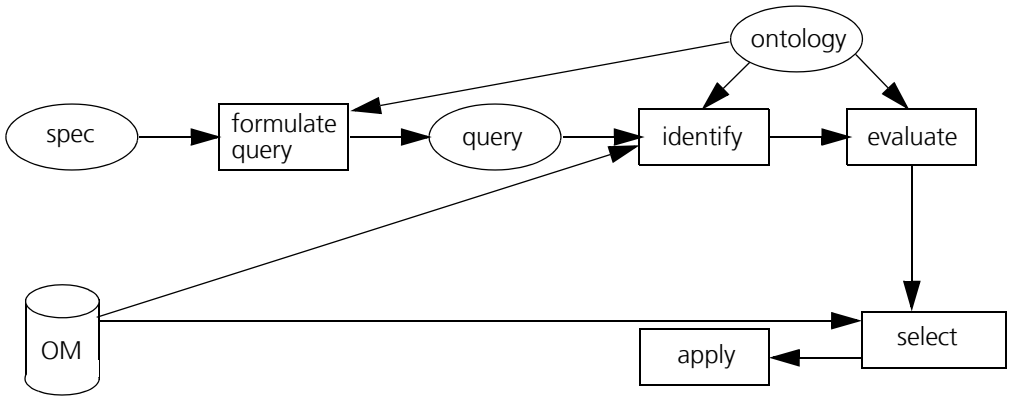

Figure 3: The »ideal«, sequential usage model of an OMMS.

In practice, this idealized process does not take place. It starts with the fact that users try to optimize the effort for maximum information gain [Har92]. This means for our usage model that the user will not formulate a query with all known information (from the specification in the user's mind), but rather specify only some attribute values (in interviews, experts stated that at most ten values

ought to suffice for a query) [Nic98]. This may result in underspecified queries which in turn lead to an unsatisfactory retrieval result. If the user thinks that the system can do better, he will go back to step »formulate query« and supply additional information and reissue the search request.

Also, it is unlikely that a user will solely select artifacts on the basis of their characterizations. Typically, he will examine the artifacts using some editor to make the decision on whether to apply them, or not (e.g., Is the artifact well documented? Can it be easily understood?). However, aim of the characterization knowledge is to limit the number of artifacts that have to be examined in this way as well as to reduce the effort needed to examine the artifacts (by supplying information that can only be extracted using large amounts of effort, e.g., correctness of a technical design with respect to its specification). After the examination of the artifact the rating of the usefulness of the artifacts may change. The deepened understanding may also lead to additional queries, starting a new usage process.

Finally, it may turn out *after* applying the artifact that it was not the best candidate for the purpose. If the task could not be performed by applying a retrieved artifact, the user may want to reissue his (possibly refined) query to find more suitable artifacts.

5 Diagnosing for Improvement

At each step of the usage model presented in the previous section, indicators may be examined to identify improvement potential. The basic idea is to diagnose situations that lead to suboptimal usefulness. These situations can be described using the variation factors of usefulness as they have been presented in Section 3. Based on the diagnosis, changes can be suggested that (hopefully) will lead to improvements of the OMMS. As we have already seen in Section 3, not all of the variation factors can be changed by purely technical means. Although some of these variation factors influence the usefulness of an OMMS substantially, the diagnosis and change of these factors is beyond the scope of this paper.

To ease the understanding of situations that can be changed by tailoring the OMMS, Figure 4 shows the usage model enhanced by change steps to the OMMS. In the following, each of the steps of the usage model is examined in detail.

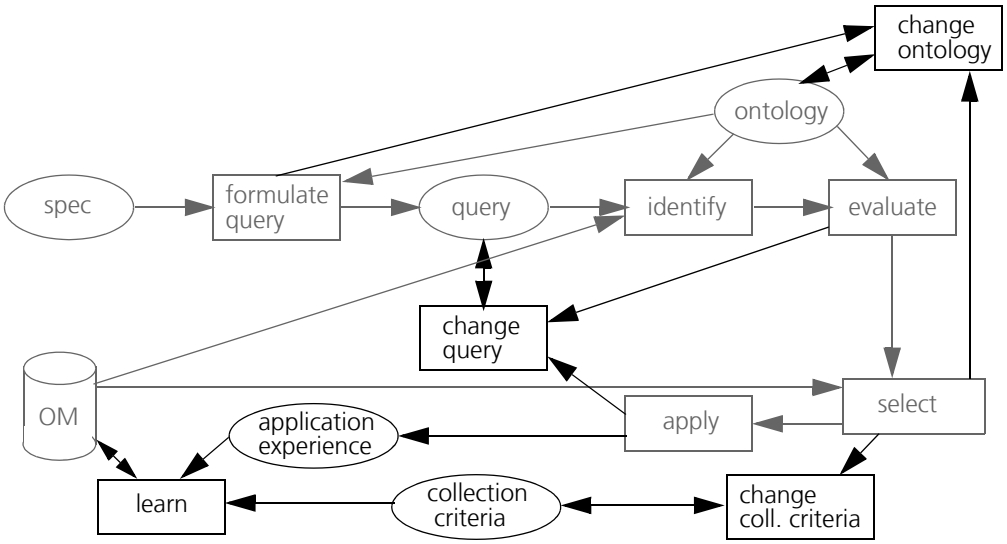

Figure 4: The usage model (shown in grey) can be enhanced by change steps to the OMMS.

- **Formulate query**. At this step it may turn out that the universe of discourse to be covered by the OM, is actually not covered. For instance, if no concept for lessons learned is part of the conceptual knowledge, it is not possible to specify lessons learned for retrieval. If lessons learned are needed by the users, they should be part of the universe of discourse of the OM. Therefore, a new concept for lessons learned should be introduced. At the same time,

existing lessons learned (e.g., in form of memos and minutes) should be analyzed and characterized for their retainment as artifacts in the OM.

- **Identify**. This step is performed without user interaction. Hence, no situation for improving the usefulness can be identified (by the user) during this step.

- **Evaluate**. This step has an automatic and a manual part. As with the step »identify«, the automatic part cannot be used for identifying improvement potential (by the user). Considering the extended usage model where the user employs some editor to examine candidate artifacts to decide on their usefulness, the manual part of the step »evaluate« is actually a three step process:

  1. Decide (based on the characterizations) which artifacts to examine in which order (artifacts that are deemed useless based on the characterizations are neglected).
  2. Examine the artifacts in the predetermined order. Stop if only one artifact is needed and an examined artifact meets the needs.
  3. Mark all useful artifacts. If not enough useful artifacts have been marked, decide whether to go back to step 1 or to reissue a revised query.

  During the first step, the user may not be able to decide whether to examine an artifact or declare an artifact as »useless«. As the examination of an artifact can require considerable effort (imagine, you have to decide whether a textbook actually contains information important to your research work), the user wants to examine only those artifacts that have a high potential of being useful. If the user is not able to make this decision, some information about the artifact is missing. This can be supplied as part of the characterization. Therefore, for all artifacts the user cannot decide on, the user should articulate the missing information. This feedback can be used to improve the conceptual knowledge of the OM in a goal-oriented way by adding new attributes. If the missing information is supplied by the OM maintenance team (i.e., including the values for new attributes), the user will be able to decide next time he issues a similar query. The usefulness of the system will have been improved.

  The output of the first step is the order in which potentially useful artifacts are examined. If the examination of the artifacts is invoked under the control of the OMMS, the OMMS can record this order during step 2. Ideally, the order should be the same as the one determined during the automatic part of the step »evaluate« (as pointed out in Section 3). If it is not or if artifacts which are placed high up in the similarity-based ordering although they were deemed useless by the user, the user can be asked why he chose a different ordering. The reasons may be either (a) underspecified queries (the user knows more than he has specified, and he matches the characterizations

with the unspecified but known information; in this case nothing needs to be improved) or (b) improper similarity functions (either because of inadequate local or global similarity functions or because of undocumented knowledge known/assumed by the user; in the latter case additional attributes should be defined capturing the undocumented knowledge and the new knowledge should be considered by the similarity functions).

- **Select**. After one or more artifacts have been examined, the best suitable artifact is selected to be applied. If the user shall not be bothered with too many questions (these will arise especially after the initial set-up of an OMMS), the questions can be restricted to the artifact actually applied. This alternative will not be able to identify situations in which artifacts were originally judged to be useful (based on the characterization), but later judged to be useless (based on the artifact's examination).

  At any rate, if the selection step yields no selected artifact, the user should be asked to give a reason why the most similar artifact was not chosen. In this case, a hole in the coverage of the OM has been identified. The artifact needed to fill this hole is both specified by the original query supplied by the user and the reason why the most similar artifact does not cover the requirements. Based on this feedback, the collection criteria for the type of the requested experience should be analyzed. Do they allow the collection of an artifact similar enough to the requested one? If not, the collection criteria should be changed. If the missing artifact is deemed to have a high application potential in the future by other users, either a separate project for creating such an artifact may be started or – if the artifact is created as part of the project the user belongs to – the artifact may be stored after the project has been completed.

- **Apply**. During the application of an artifact, experience such as effort for understanding and modifying the artifact as well as how the artifact should (not) be changed should be recorded. Such experience can then be attached (in form of an extended characterization) to the artifact after its application. In this way, the applicability information is improved continuously with each application of an artifact.

  During the artifact's application, it may turn out that the artifact is not as useful as originally estimated. If this is the case, one of four choices can be made:

  1. Ignore the fact and continue applying.
  2. Stop applying and do nothing more (e.g., if it turns out that a lesson learned is not applicable in the current situation).
  3. Stop applying and create the needed artifact from scratch (e.g., a project schedule).

4. Stop applying and retrieve another (hopefully more useful) artifact (e.g., a project schedule).

In the latter case, the old query may serve as an entry point.

The descriptions above show how improvements in the conceptual knowledge and contents of an OM can be pinpointed by automatically collecting data in the form of protocol cases and asking the user for feedback if the system behaves not as expected (based on the usage model). Changes regarding other branches of Figure 2 (e.g., user interface and missing experience items the user is aware of) can be suggested based on answers to questions posed to the user after each (or after each n-th) retrieval attempt at the end of step »select«. However, in contrast to the situations outlined above, these questions cannot be posed based on some (automatic) analysis and, thus, must be posed unconditionally. This might lead to an overburden on the user who has to devote his time in answering the posed questions.

Finally, the variation factors listed in the »user« branch deserve a closer look. While the precision and the correctness of the query has been taken care of by allowing the user to reissue a revised query, the variation factor »retrieval goal« may not be underestimated. Different retrieval goals have different information needs and the usefulness of the retrieval result is measured (at least partly) in terms of how well the presented experience items are suited to perform some predefined task. Consequently, both the information need (represented by a set of attributes) and the similarity functions (estimating the usefulness) may differ. Therefore, all optimizations of the OMMS must be performed with respect to the retrieval goal. Otherwise, an improvement done for one retrieval goal may result in a change to the worse for another retrieval goal.

As a consequence, an OMMS should ask the user for his retrieval goal at the beginning of the session. Based on the retrieval goal, the query attributes can be restricted to the relevant ones. Thus, the user is guided more effectively which in turn also helps to reduce the risk of incorrect and/or meaningless queries. In addition, optimal similarity functions can be selected. Finally, the retrieval goal can be used to tailor the OM for specific retrieval goals – the optimizations do not influence the behavior of the OMMS for other retrieval goals. However, besides the (marginal) additional effort on part of the user to specify the retrieval goal, a new error source is introduced. Now, not only the query may be incorrect but also the specification of the retrieval goal. Hence, it may not be enough for the user to change his query in case the system does not behave as wanted, but perhaps the user must correct his retrieval goal as well. An empirical investigation is needed to find out whether an average user is capable of recognizing what to change.

6 Improvement With Protocol Cases – An Example

The representation of all the information being collected during the OM usage process in the form of protocol cases (see Figure 5) helps the OM maintenance team in systematically analyzing the information, because they can apply CBR based on the formally described parts of a protocol case. During this analysis user problems may be identified and the OM maintenance team could directly contact the respective user. Also based on the analysis the contents of the OM may be improved. In addition, the behavior of the OMMS may be improved (see Figure 6)

- manually by the OM maintenance team and/or
- by adding revised, qualified protocol cases to the system such that the user can access them and/or
- by applying CBR for the automatic interpretation of these cases by the OMMS.

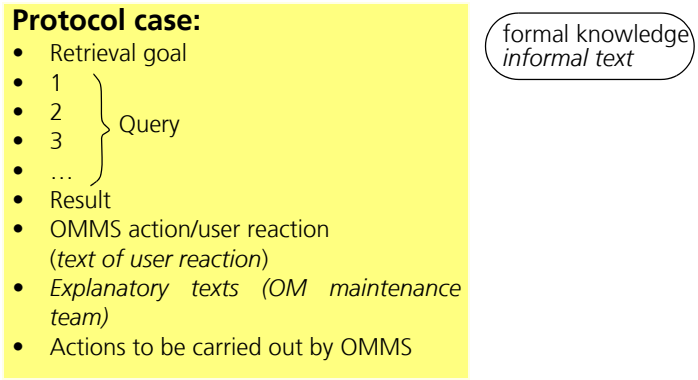

Figure 5: A protocol case contains a log of the human-computer interaction as well as actions aiming at the improvement of the organizational memory.

In the following, we will demonstrate the utilization of protocol cases using GQM plans as an example. GQM (Goal/Question/Metric Paradigm) [BDR96] is an innovative technology for goal-oriented software engineering measurement [GB97]. GQM helps defining and implementing operational and measurable software improvement goals. It has been successfully applied in several companies, such as NASA-SEL, Bosch, Digital, and Schlumberger [CEM96]. In GQM programs, the analysis task of measurement is specified precisely and explicitly by detailed measurement goals, called GQM goals. Relevant measures are derived in a top-down fashion based on the goals via a set of questions and quality/resource models. This refinement is precisely documented in a GQM plan, providing an explicit rationale for the selection of the underlying measures.

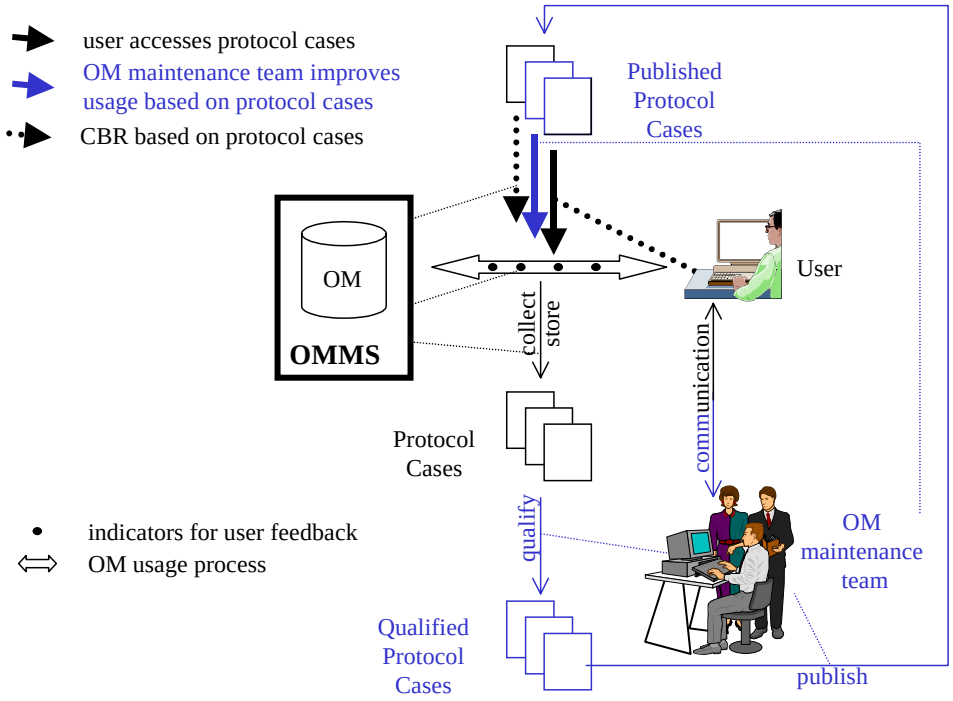

Figure 6: Protocol cases are the basis for improving the organizational memory.

GQM-Based Measurement: GQM Plan
A GQM plan is developed based on a measurement goal consisting of the following components [BDR96]:
- a goal, defining the object, purpose, quality focus, viewpoint and the context of the measurement program,
- a set of questions, operationalizing the goal,
- a set of models, specifying how to answer the questions,
- a set of measures, operationally defining the data to be collected to feed the models.

GQM Goal: Analyze the Software process in order to characterize relaibility from the viewpoint of the software developer in the company Y.

Q_1 What is the overall number of failures reported before delivery?

M_1.1 count of failure reports turned in before delivery [ratio: integer][1]

Q_2 What is the distribution of failures reported before delivery by criticality level?

Model: Distribution = (# critical failures/ total # failures, # uncritical failures/ total # failures)
critical: complete breakdown of system; uncritical: unable to perform one or more of the functions F1-F6

M_2.1 classification by criticality [ordinal:uncritical;critical]

M_2.2 count of failure reports before delivery [ratio: integer]

Q_3 What is the distribution of faults by life cycle phase of detection before delivery?

Model: Distribution = (# faults in REQ/ total # faults,# faults in HLD/ total # faults, # faults in LLD/IMP/ total # faults)

M_3.1 count of fault per life cycle phase where the fault was introduced [nominal: REQ, HLD, LLD/IMP]

Q_4 What is the total rework effort?

Model: rework effort = (effort to isolate fault + effort to correct fault)

M_4.1 for all failures reported before delivery: effort to isolate the faults that caused the failures (person-hours) [ratio: integer]

M_4.2 for each fault detected before delivery: effort to correct the fault (person-hours) [ratio:integer]

1[scale: range]

Figure 7: Simplified example of a GQM plan (taken from [TG99]).

The data collected is interpreted in a bottom-up fashion considering the limitations and assumptions underlying each measure. The process of planning GQM programs can be substantially supported through the reuse of measurement experiences [GB97, GABT98]. The GQM plan is a principal element of the planning of a GQM-based measurement program (see Figure 7).

The contents of a GQM plan can be (partially) characterized by the measurement goal as indicated in Figure 7. For our example, we will assume that the organizational memory consists of the GQM plans listed in Table 1 and the project characterizations listed in Table 2..

Name	MObject	MPurpose	Quality focus	MViewpoint	Context
A:X - Adaptability 1	Design document	Characterization	Adaptability	Software developer	A: X
A:X - Adaptability 2	Requirements document	Characterization	Adaptability	System engineer	A: X
A:X - Completeness	Requirements document	Characterization	Completeness	System engineer	A: X
A:X - Efficiency 1	Design inspection	Characterization	Efficiency	Project supporter	A: X
A:X - Efficiency 2	Code inspection	Characterization	Efficiency	Project supporter	A: X
A:X - Effort	Development process	Characterization	Effort	Technical leader	A: X
A:X - Reliability	Software product	Characterization	Reliability	Quality assurer	A: X
B: Y - Communication efficiency	Communication	Characterization	Communication efficiency	Software developer	B: Y
B: Y - Comprehensibility	Development document	Characterization	Comprehensibility	Software developer	B: Y
B: Y - Effectiveness	Quality assurance activities	Characterization	Effectiveness	Department head	B: Y
B: Y - Reliability	Software product	Characterization	Reliability	Software developer	B: Y
C: Z - Effort	Development process	Characterization	Effort distribution	Technical leader	C: Z
C: Z - Usability	Specification	Characterization	Usability	Software developer	C: Z
C: Inspection - Effort	Inspection process	Characterization	Effort	Department head	C: Z
D: W - Effort 1	Development process	Characterization	Effort	Technical leader	D: W
D: W - Effort 2	Development process	Characterization	Effort	Department head	D: W
D: W - Effort 3	Development process	Characterization	Effort	Software developer	D: W
D: W - Reliability 1	Development document	Characterization	Reliability	Software developer	D: W
D: W - Reliability 2	Development document	Characterization	Reliability	Technical leader	D: W
D: W - Reliability 3	Development document	Characterization	Reliability	Department head	D: W

Table 1: GQM plans of the organizational memory

Project	Duration	Team size	…
A: X	24	100	
B: Y	<unknown>	200	…
C: Z	18	20	…
D: W	8	7	…

Table 2: Project characterizations of the organizational memory

A query for a GQM plan that describes the measurement of the effort of a software process (includes both the development process and the quality assurance activities) from the viewpoint of a project manager for a project with a duration of 6 months and a team size of 3 would result in a protocol case as shown in Table 3. The protocol case reveals:

- The user changed the query once. He extended it by a specification of the context. Apparently, the query was too imprecise.
- »Text of user reaction (1)« shows two problems:
  1. The user has a hard time to decide which viewpoint (department head or technical leader) is appropriate for the needed viewpoint
  2. Information about the completeness of GQM plans is missing in the characterization of GQM plans. As it is unclear what is meant exactly by »completeness of GQM plans« the maintenance team will have to talk to the user before adding a new attribute to the characterization schema of GQM plans.
- Because there is no GQM plan on the software process, but only on its parts (development process and quality assurance activities), the user decides to merge two existing GQM plans (positions 1 and 4 are checked out). This fact is emphasized by »Text of user reaction (3)«.
- The deviation from the expected user reactions (viewing position 4 before position 6 and checking out positions 1 and 2) does not require a change of the OMMS because the user grouped the returned positions by their objects (see »Text of user reactions« (2) and (3)).
- The last three sections are empty because they are filled out by the maintenance team.

The maintenance team reviews this protocol case as part of its regular analysis activities. The team:

1. asks U. Ser what he means by completeness of GQM plans and adds a new attribute »completeness« for GQM plans. The »completeness« values of existing GQM plans are set to <unknown>; however, the attribute value will be determined for all future GQM plans stored in the OM.

Section	Dimension	Value
User	Name	U. Ser
Retrieval goal	Object	GQM plan
	Purpose	Modification
	Viewpoint	Project supporter
Query (1)	MObject	Software process
	MPurpose	<undefined>
	Quality focus	Effort
	MViewpoint	Project manager
	Context	<undefined>
Query (2)	MObject	Software process
	MPurpose	<undefined>
	Quality focus	Effort
	MViewpoint	Project manager
	Context: Duration	6
	Context: Team size	3
	Context: …	<undefined>
Result	Position 1	D: W - Effort 1 (0.976)
	Position 2	D: W - Effort 2 (0.976)
	Position 3	C: Z - Effort (0.967)
	Position 4	C: Inspection - Effort (0.967)
	Position 5	D: W - Effort 3 (0.965)
	Position 6	A: X - Effort (0.962)
	Position 7	A: X - Efficiency 1 (0.896)
	Position 8	A: X - Efficiency 2 (0.896)
	Position 9	D: W - Reliability 2 (0.889)
	Position 10	D: W - Reliability 3 (0.889)
OMMS action/user reaction	User reaction (1)	View position 1
	User reaction (2)	View position 2
	User reaction (3)	View position 3
	User reaction (4)	View position 6
	User reaction (5)	View position 4
	User reaction (6)	View position 7
	OMMS action (1)	Ask why user could not decide on usefulness without viewing the GQM plans
	Text of user reaction (1)	Answer: »(a) it was not clear whether the department head or the technical leader viewpoint matches the needed project manager perspective better; (b) there was no information on how complete the GQM plans were«
	OMMS action (2)	Ask why position 6 is preferred over position 4
	Text of user reaction (2)	Answer: »Positions 1-3 and 6 are about the development process, whereas positions 4 and 7 are about the inspection process.«
	User reaction (8)	Check out position 1
	User reaction (9)	Check out position 4
	OMMS action (3)	Ask why position 4 is preferred over positions 2 and 3
	Text of user reaction (3)	Answer: »In contrast to position 4, positions 2 and 3 do not cover quality assurance activities.«
Explanatory texts	–	–
Actions to be carried out by OMMS	–	–
Suggestions to the user	–	–

Table 3: Exemplary protocol case produced by a query

2. seeks a solution for deciding which viewpoint is best for the project manager. It does so by modifying the protocol case shown in Table 3:
   - Removing query 1
   - Defining an »=« filter for »MViewpoint« (meaning that this protocol case will only be retrieved if the user needs a GQM plan for the project manager; however, »MObject« and »Quality focus« may be similar).
   - Adding »There was no GQM plan available for the project manager« as explanatory text
   - Adding »A GQM plan for the technical leader was used which corresponded better to the needs of the project manager than the ones for the department head« as suggestions to the user.
   - Adding the action »ask user« as an action to be carried out by the OMMS with the following prompt: »Hypothesis: Whether a GQM plan from the viewpoint of the department head should be preferred over a GQM plan from the viewpoint of the technical leader depends on whether the department head typically acts as the project manager. For organizations in which the line and project managers are not the same, the project manager is more similar to the technical leader than the department head.
     (a) Would you agree with this?
     (b) If not: why not?
     (c) What other factors influence the similarity between the project manager and the department head/technical leader?«

The latter OMMS action exemplifies how the users can be involved *actively* in the improvement of the OMMS while they are trying to solve a similar problem themselves. Alternatively, the users could be called and asked for their opinion (e.g., as members of a steering committee). However, in this case they would be asked without having a concrete problem at hand. Therefore, the quality of the answer is likely to be lower than if they have to solve a similar problem anyhow. In the particular example presented, the user will be asked only if he looks for a GQM plan for a project manager that has a similar measurement object, quality focus and context. The OMMS can ensure that each user is asked each question at most once.

7 Conclusion

In this paper we have presented a method for improving the perceived usefulness of an organizational memory incrementally through user feedback. It is based on a general usage model, a cause-effect-model for usefulness as perceived by the user, a set of indicators for improvement potential, and an organizational level case-based reasoning approach (based on protocol cases) that is flexible concerning the degree of using fully automated interpretation of the cases by the organizational memory management system or manual case analysis and interpretation by the organizational memory maintenance team. The organizational memory improvement method considers the practical constraints typically encountered in industrial environments, e.g., limited time of users. Currently both our general organizational memory approach as well as the improvement method are validated within a number of industrial and in-house projects. It is expected that the method will lead to quick improvements after the introduction or extension of an organizational memory. Later on, the change rate will decrease. Thus, the additional effort requested from the users for their feedback will be limited in time.

Acknowledgements

The authors would like to thank Christiane Gresse von Wangenheim for providing the exemplary GQM plan.

References

[AA96] Klaus-Dieter Althoff and Agnar Aamodt. Relating case-based problem solving and learning methods to task and domain characteristics: Towards an analytic framework. *AICom - Artificial Intelligence Communications*, 9(3):109–116, September 1996.

[ABT98] Klaus-Dieter Althoff, Frank Bomarius, and Carsten Tautz. 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, August 1998.

[ADK98] Andreas Abecker, Stefan Decker, and Otto Kühn. Organizational memory (in German). *Informatik-Spektrum*, 21(4):213–214, August 1998.

[Alt97] Klaus-Dieter Althoff. Evaluating case-based reasoning systems: The Inreca case study. Postdoctoral thesis (Habilitationsschrift), University of Kaiserslautern, 1997.

[AR99] Klaus-Dieter Althoff and Michael M. Richter. Similarity and utility in non-numerical domains. In W. Gaul and M. Schader, editors, *Mathematische Methoden der Wirtschaftswissenschaften*. Physica-Verlag, Heidelberg, Germany, 1999.

[AW91] K.-D. Althoff and S. Wess. Case-based knowledge acquisition, learning and problem solving in diagnostic real world tasks. *Proceedings of the Fifth European Knowledge Acquisition for Knowledge-Based Systems Workshop*, pages 48–67, 1991.

[BB91] Bruce H. Barnes and Terry B. Bollinger. Making reuse cost effective. *IEEE Software*, 8(1):13–24, January 1991.

[BCR94] Victor R. Basili, Gianluigi Caldiera, and H. Dieter Rombach. Experience Factory. In John J. Marciniak, editor, *Encyclopedia of Software Engineering*, volume 1, pages 469–476. John Wiley & Sons, 1994.

[BDR96] Lionel C. Briand, Christiane M. Differding, and H. Dieter Rombach. Practical guidelines for measurement-based process improvement. *Software Process*, 2(4):253–280, December 1996.

[BR88] Victor R. Basili and H. Dieter Rombach. The TAME Project: Towards

improvement–oriented software environments. *IEEE Transactions on Software Engineering*, SE-14(6):758–773, June 1988.

[CEM96] The CEMP Consortium. Customized establishment of measurement programs. Final report, ESSI Project Nr. 10358, Germany, 1996.

[Coh89] P. R. Cohen. Evaluation and case-based reasoning. In K. Hammond, editor, *Proceedings of the Second DARPA Workshop on Case-Based Reasoning*, pages 168–172. Morgan Kaufman, 1989.

[Coo97] William S. Cooper. On selecting a measure of retrieval effectiveness. In K.S. Jones and P. Willet, editors, *Readings in Information Retrieval*, pages 191–204. Morgan Kaufmann Publishers, 1997.

[Dam98] Leela Damodaran. Development of a user-centered IT strategy: A case study. *Behavior and Information Technology*, 17(3):127–134, 1998.

[GABT98] Christiane Gresse von Wangenheim, Klaus-Dieter Althoff, Ricardo M. Barcia, and Carsten Tautz. Evaluation of technologies for packaging and reusing software engineering experiences. Technical Report IESE-Report No. 055.98/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern (Germany), 1998.

[GB97] Christiane Gresse and Lionel Briand. Requirements for the Knowledge-Based Support of Software Engineering Measurement Plans. In *Proceedings of the Ninth International Software Engineering and Knowledge Engineering Conference (SEKE'97)*, pages 559–568, Madrid, Spain, June 1997.

[GKP+83] J. Gaschnig, P. Klahr, H. Pople, E. Shortliffe, and A. Terry. Evaluation of expert systems: Issues and case studies. In F. Hayes-Roth, D.A. Waterman, and D.B. Lenat, editors, *Building Expert Systems*, pages 241–282. Addison-Wesley, Reading, Mass., USA, 1983.

[GXG98] Avelino Gonzales, Lingli Xu, and Uma Gupta. Validation techniques for case-based reasoning systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(4):465–477, July 1998. Part A: Systems and Humans.

[Har92] Stephen P. Harter. Psychological relevance and information science. *Journal of the American Society for Information Science*, 43(9):602–615, October 1992.

[Hen97] Scott Henninger. An evolutionary approach to constructing effective software reuse repositories. *ACM Transactions on Software Engi-*

neering and Methodology, 6(2):111–140, April 1997.

[HS93] B. Henderson-Sellers. The economics of reusing library classes. *Journal of Object-Oriented Programming*, (4):43–50, 1993.

[JK98] Matthias Jarke and Ralf Klamma. Innovation based on computer-aided failure management: Results of the BMBF project FOQUS (in German). In *Proc. Wirtschaftsinformatik*, 1998.

[Kir94] S. Kirchhoff. *Mapping Quality of Knowledge-Bases Systems: A Methodology for Evaluation (in German)*. Josef Eul Verlag, Bergisch-Gladbach, Germany, 1994.

[Lim96] Wayne C. Lim. Reuse economics: A comparison of seventeen models and directions for future research. In Murali Sitaraman, editor, *Proceedings of the Fourth International Conference on Software Reuse*, pages 41–50, Orlando, Florida, USA, April 1996. IEEE Computer Society Press.

[MD97] Michael G. Morris and Andrew Dillon. How user perceptions influence software use. *IEEE Software*, 14(4):58–64, July/August 1997.

[Nic98] Markus Nick. Implementation and Evaluation of an Experience Base. Diploma thesis, Fraunhofer IESE, University of Kaiserslautern, 1998.

[NT98] Markus Nick and Carsten Tautz. Practical evaluation of an organizational memory using the goal-question-metric technique. Technical Report IESE-Report No. 063.98/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern (Germany), 1998.

[NT99] Markus Nick and Carsten Tautz. Practical evaluation of an organizational memory using the goal-question-metric technique. In *Proceedings of the Workshop on Knowledge Management, Organizational Memory and Knowledge Reuse during Expert Systems '99 (XPS-99)*, Würzburg, Germany, March 1999.

[Pfe96] T. Pfeifer, editor. *Knowledge-Based Systems in Quality Management (in German)*. Springer-Verlag, 1996.

[PZ96] T. Pfeifer and T. Zenner. Using experience during failure analysis - the application of case-based techniques (in German). In R. Grob and J. Spiekermann, editors, *Workshop at the Third German Conference on Expert Systems*, pages V1–V12. Technical Report LSA-95-04, University of Kaiserslautern, 1996.

[SM83] Gerard Salton and Michael J. McGill. *Introduction to Modern Infor-*

mation Retrieval. McGraw-Hill Book Co., New York, 1983.

[SW88] J. R. Slagle and M. R. Wick. A method for evaluating candidate expert system applications. *AI Magazine*, 9(4):44–53, 1988.

[Tec99] CBR-Works. URL http://www.tecinno.de/english/products/cbrw_main.htm, 1999. tecInno GmbH, Germany.

[TG98] Carsten Tautz and Christiane Gresse von Wangenheim. REFSENO: A representation formalism for software engineering ontologies. Technical Report IESE-Report No. 015.98/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern (Germany), 1998.

[TG99] Carsten Tautz and Christiane Gresse von Wangenheim. A representation formalism for supporting reuse of software engineering knowledge. In *Proceedings of the Workshop on Knowledge Management, Organizational Memory and Knowledge Reuse during Expert Systems '99 (XPS-99)*, Würzburg, Germany, March 1999. http://www.aifb.uni-karlsruhe.de/WBS/dfe/xps99.proc.htm.

[vW96] Bert van Wegen. *Impacts of KBS on Cost and Structure of Production*. PhD thesis, 1996.

[Wes95] S. Wess. *Case-Based Reasoning in Knowledge-Based Systems for Decision Support and Diagnostics (in German)*. PhD thesis, University of Kaiserslautern, 1995.

Document Information

Title:	Improving Organizational Memories Through User Feedback
Date:	February 1, 1999
Report:	IESE-004.99/E
Status:	Final
Distribution:	Public