Information Systems Frontiers 2:3/4, 349–367, 2000

Knowledge Management for Building Learning Software Organizations

Klaus-Dieter Althoff, Frank Bomarius, and Carsten Tautz
Fraunhofer Institute for Experimental Software Engineering (IESE), Department of Systematic Learning and Improvement, Sauerwiesen 6, D-67661 Kaiserslautern, Germany
E-mail: {althoff, bomarius, tautz}@iese.fhg.de

Abstract. *Due to the steadily increasing demands of the market, strategic management of knowledge assets, or learning organizations, are becoming a must in industrial software development. This paper presents work done at Fraunhofer IESE, where learning organizations for software development organizations are being developed and transferred into industrial practice. It describes how learning organizations for the software domain can be built upon both mature approaches from Software Engineering like the experience factory model and industrial strength technology from knowledge management. A system to support the learning software organization is sketched and experiences regarding the implementation of this system and learning software organizations in general are presented.*

Key Words. *knowledge management, software domain, learning software organization, knowledge representation, intelligent retrieval and storage system, software engineering experience environment*

1. Introduction

The demands in today's software industry, such as short lead-time, frequent introduction of new technologies, increasing application complexity, and increasing quality requirements, are among the toughest to be found in industry. Traditional production-oriented approaches to meet these demands, like quality assurance or statistical process control, fall short or are just not applicable in the development-oriented software domain. In such an environment, continuous fast learning is one of the top priority requisites to acquire and maintain leading-edge competencies. Traditional individual or group learning, as a means of adapting to new demands or of adopting new methods and techniques, is often far too slow and ineffective. This is especially true if it is not pursued in a goal-oriented way, managed as a project crucial to a company's success, and supported by organizational, methodical, and technical means. So, learning on an organizational level and capitalizing on an organization's knowledge assets becomes imperative for modern software-dependent industries. Such learning needs can be addressed by systematic application of organizational learning (OL) principles, supported by organizational memories (OM). We believe that learning organization (LO) principles will soon establish themselves as best practices. Therefore, we see a strong need to spell out OL procedures and methods that work in practice and also a need for comprehensive tool support. This paper is about the approach taken by the authors to do so for the software domain.

We see the subject matter as being composed of several dimensions:

- the processes, methods, techniques of how to implement OL in the application domain;
- the tools that support OL for that domain; and
- the organizational and cultural aspects of introduction and performance of OL.

From our experience we know that the latter one is of paramount importance for the success of a technology transfer project like, for instance, the introduction of OL (Kotter, 1996; Senge, 1990).

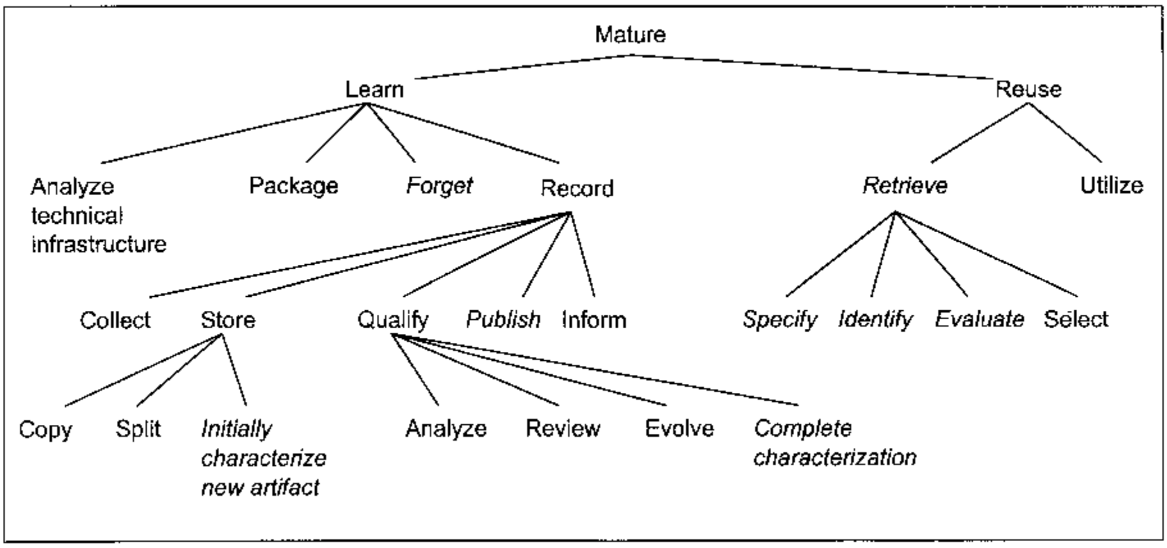


Fig. 2. *Tasks for incremental, continuous learning.*

The paper ends with projects validating our approach (Section 5) and an outlook (Section 6).

2. Sample Scenario

To exemplify the tasks of an EB system we use a scenario that describes some typical project management activities and how these activities can be supported by an EB. In particular, the scenario shows how:

- a software development project is planned and performed using experiences from past projects provided by the EB;
- a software development organization learns, i.e., how the contents of the EB is enhanced and restructured according to new project experience.

2.1. Simplified structure model for the experience base

Before the mechanisms for reusing and learning SE knowledge can be explained, the structure model of the EB must be presented. The structure model of the EB can be seen in analogy to the data models of database management systems. It guides the user while he retrieves or stores knowledge.

The EB shall contain several types of knowledge each represented by a separate concept. Every instance in the EB is described by exactly one of these concepts. During retrieval, one of the concepts is used as a template to be filled in, in order to specify the knowledge to be searched for. This implies that the user has to know the type of knowledge he needs when he specifies a query. The type of knowledge can be regarded as a filter: only instances described by the selected concept will be searched (task "identify").

For the scenario described here, the structure model shown in Fig. 3 is used. The meaning of the different concepts is described in Table 1. Each concept has two kinds of attributes: terminal and nonterminal attributes. Terminal attributes model how SE entities are specified for storage and retrieval, whereas nonterminal attributes model semantic relationships. Nonterminal attributes are implemented using references. All semantic relationships are bidirectional. This is indicated in Fig. 3 by the arcs between the concepts. For example, the object of a "quantitative experience" can be a "technique", "process model", or "product model". The context of a "quantitative experience" is described by a "project characterization". Vice versa, all "quantitative experience" gained in a project can be found by following the references stored in the nonterminal attribute "quantitative exp" of a "project characterization".

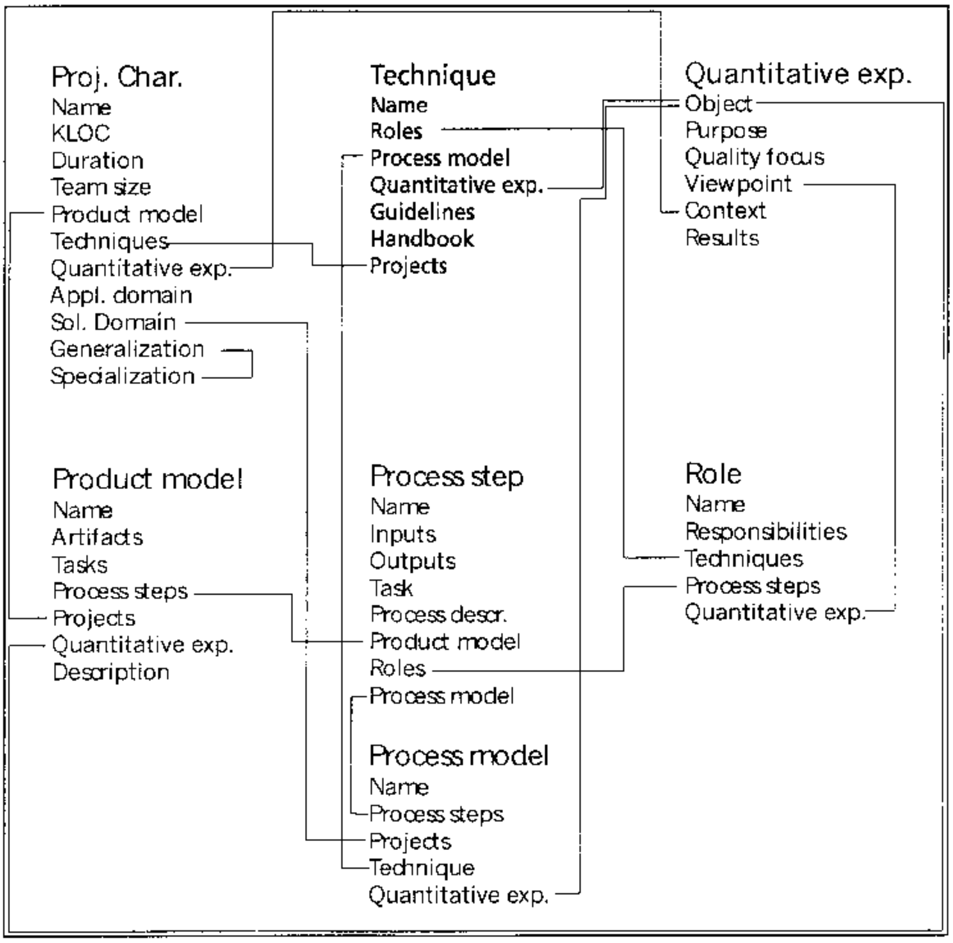


Fig. 3. Simplified structure model of an exemplary EB.

Table 1. *Main concepts of the exemplary EB*

Process model	Specifies in which order which process steps are performed.
Process step	An atomic action of a process that has no externally visible substructure.
Product model	Defines the structure of software development products as well as the tasks to be performed. It does not describe, however, how to perform these tasks (described by the corresponding process step) nor in which order the tasks are to be performed (described by the process model).
Project characterization	Summarizes the relevant characteristics of a project. It contains applicability conditions for most other types of SE knowledge.
Quantitative experience	A pair consisting of a measurement goal and the results of the measurement. The measurement goal is always defined at the beginning of a project using five facets: the object to be analyzed, the purpose for measuring, the property to be measured (quality focus), the role for which the data is collected and interpreted (viewpoint), and the context in which the data is collected. The data collected and interpreted is only valid within the specified context.
Role	A set of responsibilities, enacted by humans.
Technique	A prescription of how to represent a software development product and/or a basic algorithm or set of steps to be followed in constructing or assessing a software development product.

2.2. Project setting

The fictive scenario described below is based on the following assumptions:

- The EF is established at department level at an automotive equipment manufacturer.
- The department has several groups; however, in the scenario only the groups responsible for the software development of "ABS" and "fuel injection" equipment are involved.
- The group "fuel injection" has just closed a contract with a car maker requiring a "design review", something the group has never done before.
- The new project is named "Maui".

2.3. Getting the project off the ground

The project manager has never managed a project with a design review before. Therefore, he needs information on projects conducted where a design review has been performed. From the requirement to perform a "design review" he deduces that the software documentation must at least contain the requirements and the design. Furthermore, the product model must allow the construction and verification of these products (according to the glossary of the organization, the "design review" is a verification of the design). He estimates that the project will run 12 months with 3–5 people working on it at any given time.

As a first step the project manager enters his knowledge in the form of a query searching for similar projects (in our scenario these are projects with roughly the same duration and team size) which (a) also employed design reviews and (b) also delivered requirements and design documents (Fig. 4). The three most promising project characterizations returned by the EB are shown in Fig. 5 (structural view). As can be seen, two projects, named "Hugo" and "Judy", have been performed using design reviews.

However, they have not been performed in the "fuel injection" group, but rather in the "ABS" group. Quite strikingly, in both cases a design inspection was performed besides the design review. A project characterization generalizing the characterizations of "Hugo" and "Judy" shows this explicitly (see generalization references in "Hugo" and "Judy" in Fig. 5).

By interviewing the project manager of "Judy", our project manager finds out that the inspections were performed for preparing the design review. The goal was to identify and eliminate as many design defects as possible before the customer takes a look at the design, thus increasing the confidence and/or satisfaction of the customer. Based on this discussion, our project manager decides to employ design inspections as well.

As the experience about inspections stems from a different application domain (i.e., ABS system development), the models available may not be valid for the application domain at hand. Therefore, it is decided (i.e., a measurement goal is set) to measure the effectiveness of inspections in this application domain, so as to extend the EB by effectiveness models for inspections in the "fuel injection" domain. Furthermore, inspections are seen

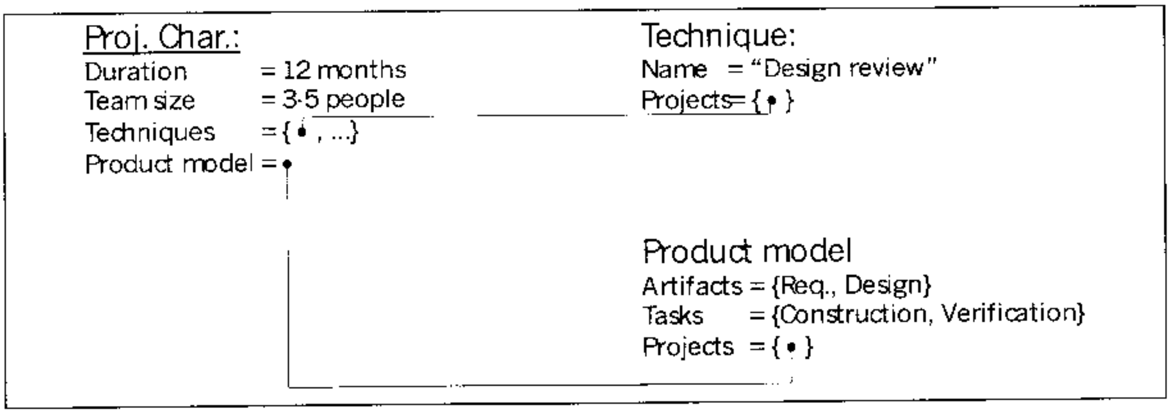


Fig. 4. Query for similar project characterizations.

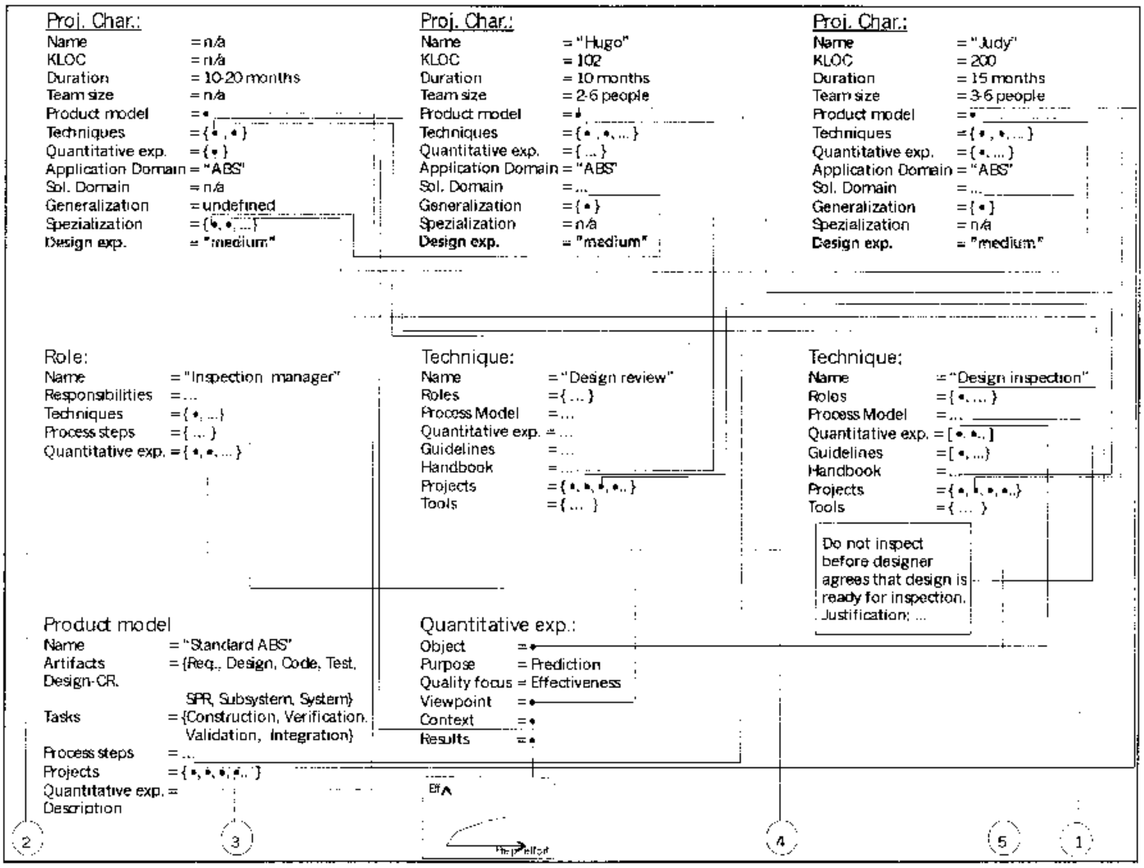


Fig. 5. Result of first query.

as a chance to improve the overall efficiency of software projects, because defects can be found earlier in the life cycle than if system tests were conducted at the end of coding and integration. It is therefore hypothesized that the rework effort for the correction of defects can be greatly reduced. To validate this hypothesis with quantitative data, a second query is formulated (Fig. 6). The query searches for quantitative experiences on efficiency that were collected on similar projects in the "fuel injection" group using the standard process model "Standard 4711", which is to be used in "Maui".

The results of the query (Fig. 7) show an efficiency range of 2.7 ± 0.4 KLOC per person month. If inspections make projects more efficient, the efficiency of "Maui" should be higher than 3.1 KLOC/PM.

As the final planning step for "Maui", the actual process models and measurement plans are being developed. The process model "Standard 4711" is taken as a basis and extended by design inspections and reviews. This results in a new process model "Standard 4711 with design insp. · review". The measurement plan used in the old projects "Vesuv" and "Purace" is tailored to the new needs, that is, the effort for performing the inspections is also considered for the computation of the efficiency. To plan the inspections, our project manager also relies on quantitative experience gained in the group "ABS" (see Fig. 5). For example, he sets the goal to achieve an effectiveness of 0.5 (typical effectiveness achieved in "ABS" projects—not shown in Fig. 5) and estimates the needed preparation effort based on this goal (see "Quantitative exp." in Fig. 5). At the same time he identifies this as a risk factor, since the model upon which these estimations are based has not been validated for "fuel injection" projects.

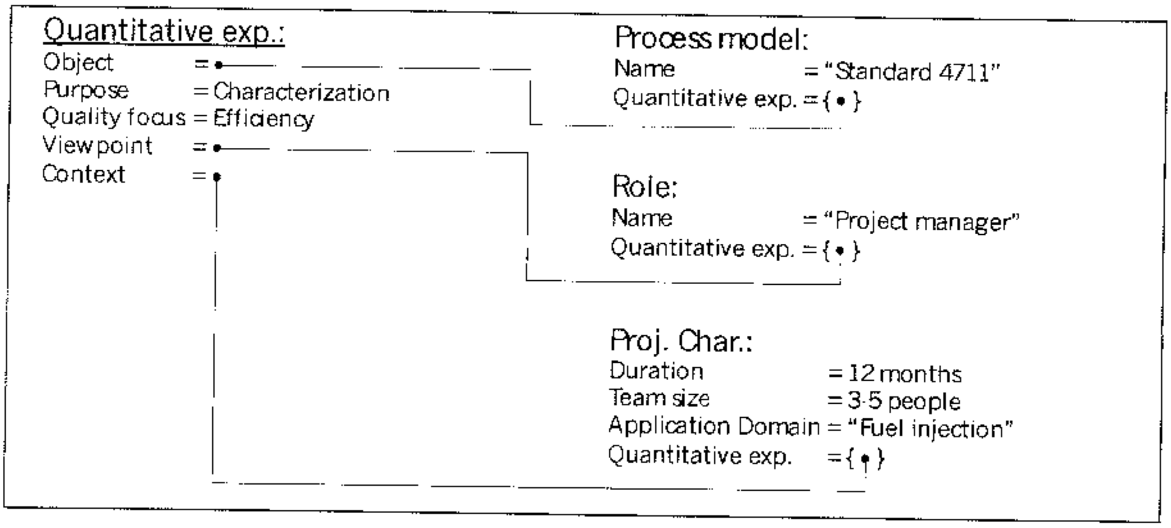


Fig. 6. Query for similar quantitative experience.

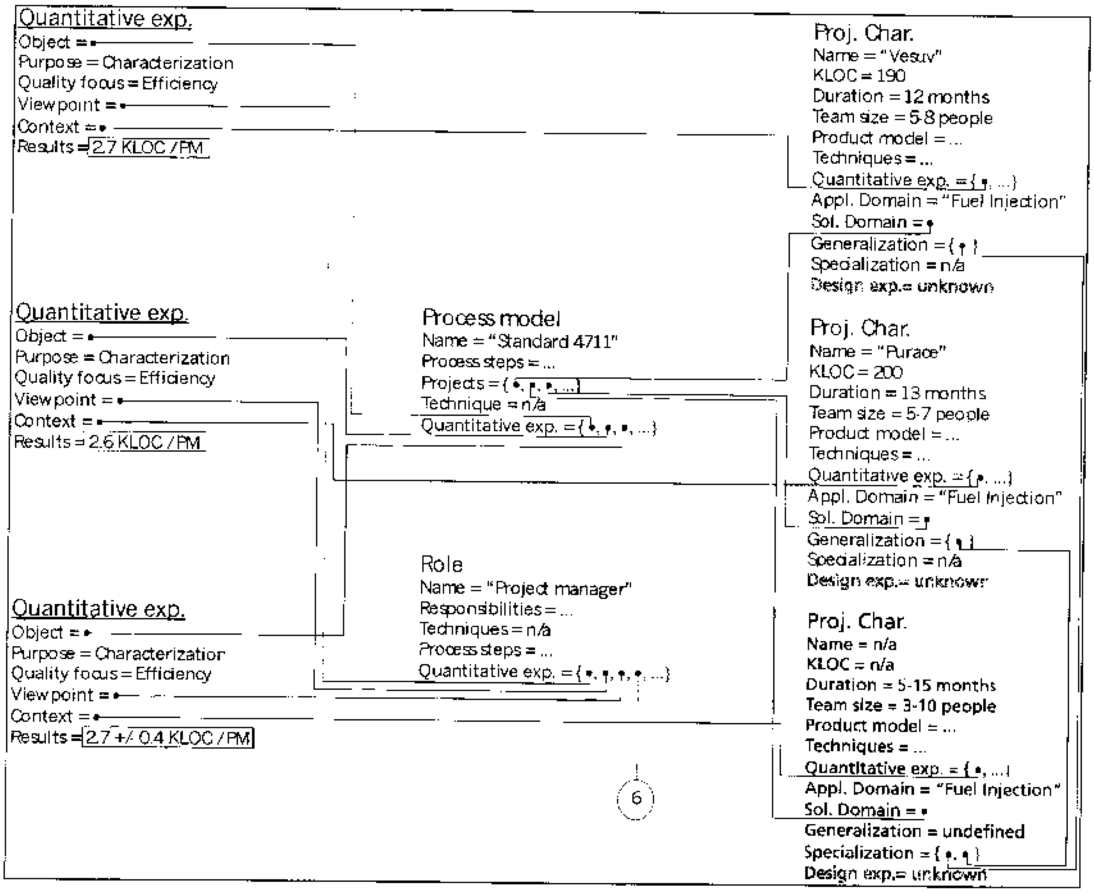


Fig. 7. Result of second query.

Looking for further risk factors, our project manager also searches the EB for guidelines associated with the techniques applied. For instance, the guideline "Do not inspect before the designer agrees that the design is ready for inspection" was found and will be employed because the justification sounds reasonable (see Fig. 5).

2.4. *Perform project*

During the performance of the project, the data for the defined measurement goals is collected. The EB may be consulted for more reusable components and problem solution statements. Detailing these reuse attempts is beyond the scope of this scenario.

2.5. *Learning from project experience*

After the project was completed, 250 KLOC had been developed. Instead of the planned 5 people, 7 people had been working on the project and project duration was prolonged by 1 month. Yet the efficiency was measured to be 3.2 KLOC/PM. However, the effectiveness of the inspections was only 0.4 instead of the planned 0.5. Therefore, further analysis was conducted showing that the experience of the designers was not considered in the model of effectiveness. In all projects conducted in the "ABS" group, the designers had a medium level of experience, whereas the designers in the "Maui" project had only little experience.

The project characterization that is the result of the post-mortem analysis, the gathered quantitative experiences, and the tailored process model "Standard 4711 with design insp. + review" become new instances of the EB. The relationships to existing instances are also specified (Fig. 8; the relationships are indicated by connectors to Fig. 5 and Fig. 7).

Since a new important applicability factor (design experience) was identified, all existing project characterizations are extended by this new attribute (see gray texts in Fig. 5, Fig. 7, and Fig. 8). For "Maui" the attribute value is "low", whereas for "Hugo" and "Judy" as well as their generalization the attribute value is "medium". For all other projects, the attribute value is set to "unknown", because it would require too much effort to collect this information. Moreover, this information would be impossible to get (at least in part), since some of the old project managers have already left the software development organization.

2.6. *Strategic aftermath*

From the inspection effectiveness and the project efficiency, no conclusive evaluation can be done with respect to the hypothesis that inspections increase project efficiency, because "Maui" could be an outlier regarding efficiency. The 3.2 KLOC/PM are quite promising, but further empirical evidence is needed. For this reason, the "fuel injection" group creates a new process model "Standard 4711 with design insp." This process model shall be applied in the next three "fuel injection" projects to be able to build a more valid efficiency model.

Since it is also expected that the inspection effectiveness will be better if more experienced designers take part, inspection effectiveness will also be measured in future projects.

2.7. *Conclusion from the sample scenario*

The scenario illustrates that:

- An EB can supply knowledge users did not expect (in the scenario, the project manager did not know that design reviews were performed in the "ABS" group).
- Goal-oriented, organizational learning leads to strategically relevant knowledge faster than learning on the level of individuals or groups (see strategic aftermath). Even if the explicitly available knowledge does not meet the current needs perfectly, it can be taken advantage of (see utilization of "design review" experience in the "ABS" group).
- SE knowledge is not static. It is complemented on a continuous basis. In the scenario, both concrete instances and structural knowledge (e.g., "design experience" is relevant for selecting the right effectiveness model) are added.

3. *Knowledge Representation*

As we have seen in the above sample scenario, in the context of software development and improvement programs many kinds of artifacts need to be stored. Exemplary kinds of artifacts are process models, product models, resource models, quality models, all kinds of software artifacts (e.g., code modules, system documentation), lessons learned about these artifacts,

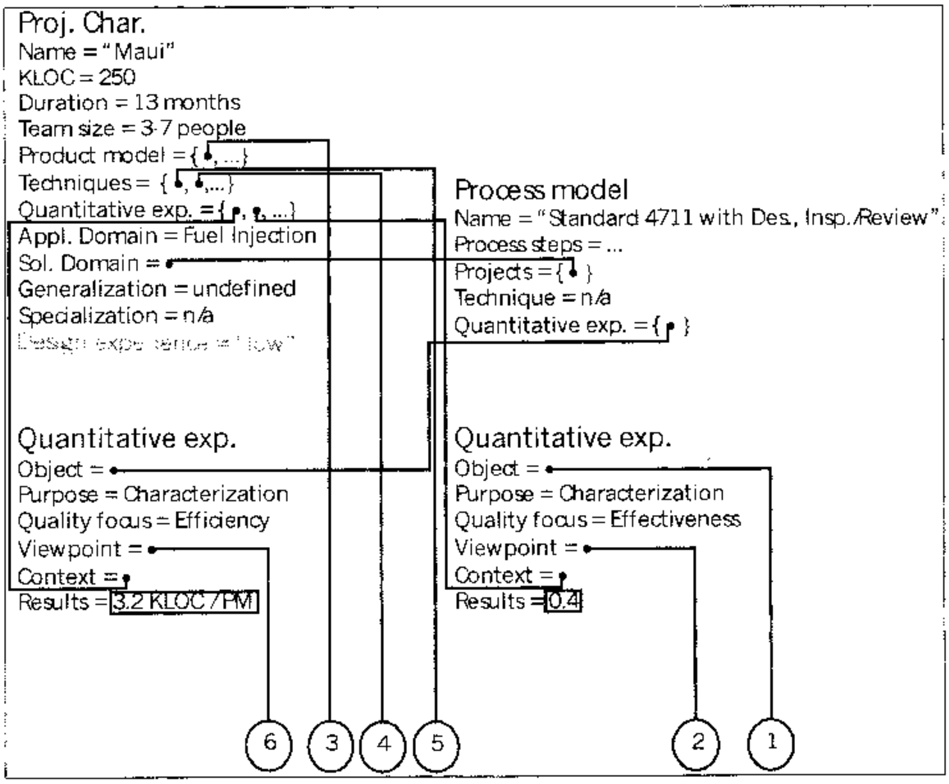


Fig. 8. Updated experience base.

and observations (results and success reports). To be able to learn and reuse effectively, all these artifacts have to be represented in a uniform way. Characterizations are a natural way to achieve this goal (Prieto-Díaz, 1991). Characterizations "extend" artifacts by complementing the information contained within the artifacts. The additional information is necessary to enable users to retrieve the most appropriate artifact. In the above scenario the project manager characterizes the project for which he is responsible. This characterization is then matched against the characterizations (instances) stored in the EB (see Fig. 5).

For an artifact to be part of the EB, it needs a characterization. This characterization is structured by the concept associated with the respective artifact, for example, a technique, a quantitative experience, a process step, etc. (see Fig. 3). Thus, all artifacts of one type are characterized using the same "characterization structure" (also called "characterization schema"). Consequently, the EB will be integrated from a heterogeneous set of information sources. Integration of the diverse artifacts into a coherent model requires a flexible, modular overall schema for the EB that allows interfacing with various tools and information sources.

All tasks that can be performed based only on the characterizations of the artifacts can be carried out generically (in Fig. 2, these tasks are printed in italics), whereas all tasks requiring to view or change the artifact itself must be performed using specialized (i.e., artifact-specific) tools. This also fulfills a very practical need of our project partners: It must be possible to integrate already existing (artifact-specific) tools. In this section, we will focus on the knowledge representation used in our EB system for the generic tasks, while the EB system architecture is subject of the next section. One important aspect to be considered when choosing an adequate representation is the way people work. When solving problems, people often do so by referring to examples of similar past problems. Such "reference" is usually through context information (e.g., "the process model that was used in the project I worked on last year"). Explicitly modeling (characterizing) a context (e.g., the project environment) of artifacts also allows to discover experience in an EB previously unknown to the user (e.g., by querying for "process models used in small projects"). Queries considering the context can also be employed for analyzing the contents of an EB, possibly resulting in generalized or aggregated experience.

Representing the context while minimizing the maintenance effort demands the characterizations to reference each other. This introduces the construct of semantic relationships, denoted as nonterminal attributes in Section 2. Once nonterminal attributes are represented explicitly, they can also be used to conduct context-sensitive retrieval. Since each software development project is different, it is very unlikely that one finds an artifact exactly fulfilling one's needs. The retrieval mechanism must therefore be able to find similar artifacts, which may then be tailored to the specific needs of the project at hand.

Similarity-based retrieval with incomplete information can also be realized based on characterizations as approaches by various researchers show. For instance, faceted classification (Prieto-Díaz, 1991) can be used to search for artifacts of one kind. Ostertag et al., (1992) extended faceted classification to also include nonterminal attributes.

In addition to the assertions provided by Ostertag's approach, we identified the need to provide value inferences (i.e., automatic value computations for attribute values that can be derived from other attribute values). This minimizes the number of attribute values that the user has to supply and thus reduces the maintenance effort. To further support the user, we use the characterization schemas to guide characterizations of both new artifacts (task "record" from Fig. 2; see also Fig. 8) and needed artifacts (task "specify"; see Fig. 4). This also mirrors the organizational separation of project organizations (dealing only with characterizations and the artifacts themselves, but not with characterization schemas) and the EF (responsible for defining characterization schemas).

These conclusions led us to the definition of the knowledge levels shown in Fig. 9.[1] To operationalize the retrieval, input, and maintenance activities, we need a formal representation of the conceptual knowledge of an EB. In the Artificial Intelligence (AI) literature, such conceptual knowledge is represented as an ontology (Kalfoglou, 1999; Landes et al., 1998; Sumner et al., 1998; Uschold and Gruninger, 1996). For SE ontologies, we use a special representation formalism named REFSENO (*re*presentation *f*ormalism for *s*oftware *en*gineering *o*ntologies; Tautz and Gresse von Wangenheim, 1999) incorporating the above mentioned representation constructs.

Fig. 9 shows a simplified ontology of an EB and an excerpt from Section 2 for the context-specific knowledge. As can be deduced from this small example, automated support for ensuring the consistency of the context-specific knowledge is needed. The generic tasks operate on the context-specific knowledge and are guided by the conceptual knowledge. For example, in Section 2 the project manager was interested in information about projects that are similar to the one he was responsible for. Thus, to retrieve project information, the EB system provided him with the schema for "project characterizations" (part of the conceptual knowledge; see Fig. 3). He then instantiated this schema by providing a characterization of the project he was responsible for (task "specify"; see Fig. 4). The EB system determines the potential instances (in this case, all stored "project characterizations"; task "identify"), computes the similarity of the potential instances to the queried one, and ranks them in descending order (task "evaluate"). Ideally, that is, if the similarity measure is defined adequately, the most similar "project characterization" is the best candidate to reuse (task "select"). However, in practice it is hard to define similarity right the first time (Henninger, 1996; Gresse von Wangenheim et al., 1999). Therefore, support for inspecting the artifact right away (in this case the "project characterization") is necessary. We will address this requirement in the next section.

For the implementation of the generic tasks we evaluated candidate approaches to implement a system that supports OL in the software domain. Case-based reasoning (CBR) technology is a promising approach (Althoff and Bartsch-Spörl, 1996; Althoff, 1997, 1999; Tautz and Althoff, 1997) to us. There are two main arguments, a technical and an organizational one, why we selected CBR as the most promising AI technology.

While the representation and reuse of software knowledge recommends an approach from the knowledge-based systems field, learning from examples suggests an approach from the field of machine learning. A technology that is rooted in both fields is CBR. It is a generalization of the faceted classification and, thus, also offers a natural (i.e., direct) solution for similarity-based retrieval based on incomplete information (as exemplified in Section 2).

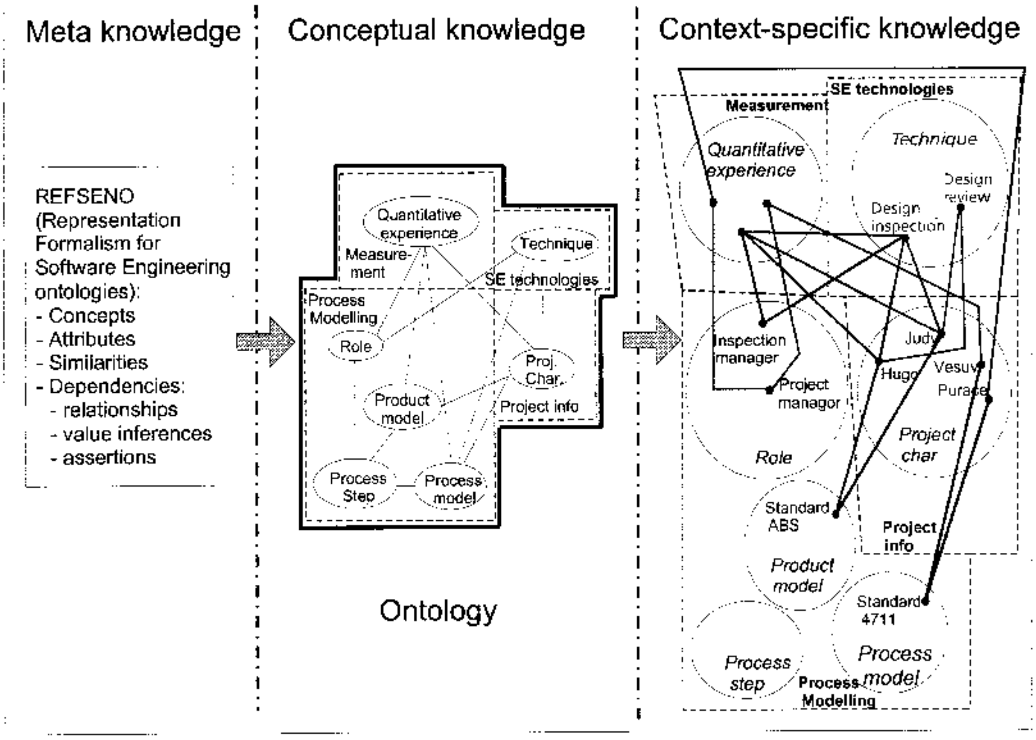
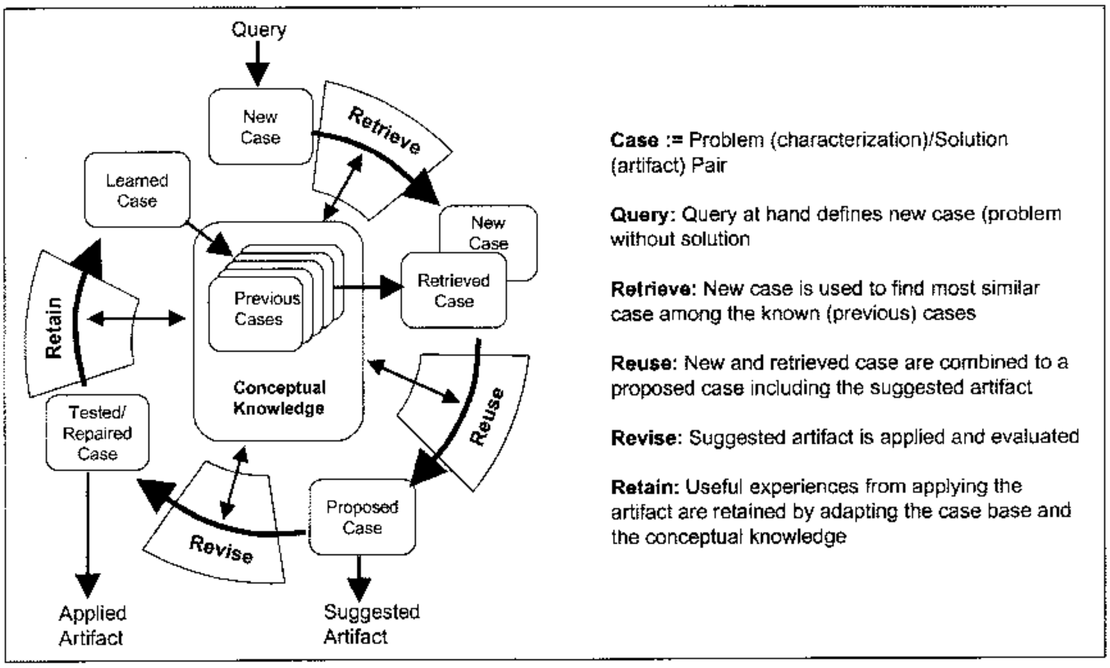


Fig. 9. *Knowledge levels of SE knowledge.*



Case := Problem (characterization)/Solution (artifact) Pair

Query: Query at hand defines new case (problem without solution

Retrieve: New case is used to find most similar case among the known (previous) cases

Reuse: New and retrieved case are combined to a proposed case including the suggested artifact

Revise: Suggested artifact is applied and evaluated

Retain: Useful experiences from applying the artifact are retained by adapting the case base and the conceptual knowledge

Fig. 10. *Case-based reasoning.*

From an organizational perspective, the tasks shown in Fig. 2 have to be supported. Already Althoff and Wilke (1997) motivated that CBR can be applied as a human-based, technologically independent strategy. That is, the basic CBR cycle (see Fig. 10) consisting of the steps "retrieve—reuse—revise—retain", as described by Aamodt and Plaza (1994), is completely carried out by humans. Viewing CBR as a human-based strategy enables the use of CBR methods for the tasks shown in Fig. 2. More details can be found in Tautz (2000).

The next section focuses on the implementation of the EB integrating the generic and artifact-specific tasks of Fig. 2.

4. System Architecture

In this section, we present a system architecture for a "software engineering experience environment" (SEEE; see Fig. 11). We distinguish between general purpose EF tools and artifact-specific application tools. General purpose tools operate on the characterizations of the artifacts (attribute values which can be searched for) in the EB, whereas the application tools operate on the artifacts themselves. Both kinds of tools act as clients using the EB server as a means for retrieving and versioning SE artifacts. The interplay of the different tools has been exemplified in the sample scenario (see Section 2). In the following we will explain parts of the tasks shown in Fig. 2 and use them to illustrate how the proposed SE experience environment works.

In the beginning of a project the general purpose search tool is started and the new project is (partially) specified guided by the characterization schemas (case models in terms of CBR tools) for project characterizations, techniques, and product models (tasks "specify" and "identify"). The search tool then returns a list of similar project characterizations (cases in terms of CBR tools) by using the CBR tool of the EB server (tasks "evaluate" and "select"). Starting from the project characterizations, the user can navigate to characterizations of relevant artifacts such as techniques employed, quantitative experiences collected, etc. The navigational search is supported through case references (links between cases), that is, for each case the user navigates to, the CBR tool is used to retrieve it.

Next, project goals (including measurement goals to enable strategic learning for the software development organization) are set based on the results of the

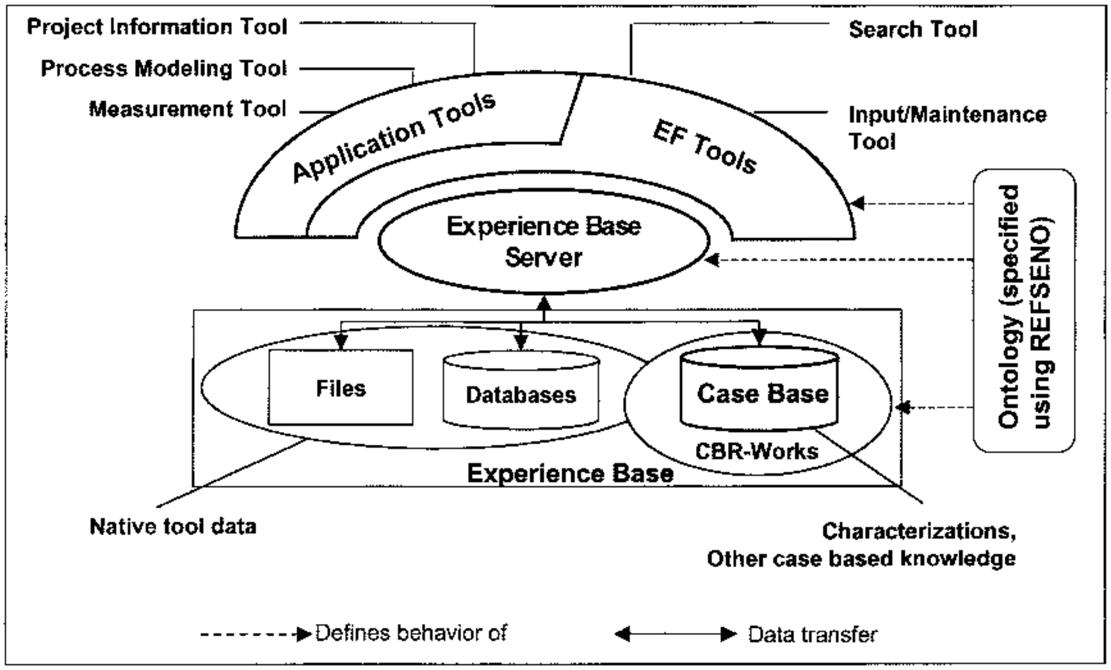


Fig. 11. *Architecture of the software engineering experience environment.*

step before. This is done in a similar manner: The search tool is invoked to find similar measurement goals from the past. However, these measurement goals and the respective measurement plans have to be adapted to project-specific needs. For this purpose, a cross-reference, which is stored as part of the artifact's characterization in the case base, is passed to the measurement planning tool. The measurement tool thus invoked loads the old measurement plan (using the data retrieval services of the EB server) and allows the project manager to tailor it to the project's needs (task "utilize"). The new measurement plan is saved using the data storage services, but its characterization is not specified yet. That is, no case is stored for it and, therefore, it is not considered as part of the EB. Thus, it is not available to other projects (yet). This avoids the usage of unvalidated artifacts.

In the same way, process models and other artifacts needed by the project are retrieved and tailored to specific needs of the project at hand. The project is then executed.

Once the project is finished, the project's artifacts (which are deemed worthwhile to keep) are stored in the EB for future projects. For this purpose, the quality of the artifacts is determined through careful analysis with the help of the application tools (task "collect"). For those artifacts to be stored (task "copy"), the export-interface of the application tools compute the attributes' values of the attribute's characterization automatically as far as possible (task "initially characterize new artifact"). This is necessary because the general purpose EF tools are not able to read the native data format of the application tools. The procedure may involve inserting several semantically related characterizations (e.g., for a code module each function may be characterized separately in addition to the characterization of the whole module). For this purpose, the artifact has to be split into chunks (task "split"). Attribute values, which cannot be computed automatically, must be entered manually. This is realized through invoking the (general purpose) input/maintenance tool, which prompts the user for the missing values. This procedure is followed for all artifacts to be stored. The newly inserted cases have to be validated and disseminated. Therefore, they are initially marked as "to be validated" at the time of their insertion. Experience engineers from the EF analyze and review the newly acquired assets to guarantee a minimal quality (tasks "analyze" and "review"). In case of minor quality deficits, the experience engineers also correct the assets by invoking the respective application tool (task "evolve"). During the review, the experience engineers also assess the reuse potential of the artifacts by using the respective application tool. As a result, the artifact may be modified to increase its reuse potential (task "evolve"). Usually this requires modification of the artifact's characterization (using the maintenance tool). In case of major quality deficits, the assets will be rejected. In case the artifacts pass the review, their characterizations are completed by the quality properties of the artifacts (task "complete characterization"). Finally, the corresponding cases are marked "validated" (task "publish") and persons interested in the new artifacts are informed (task "inform").[2] After this, it is possible for other projects to access the new artifact.

For implementing the architecture presented above, we use commercially available software, namely CBR-Works from tec:inno, Kaiserslautern (CBR-Works, 2000), as the CBR tool within the EB server. The application and EF tools are built using the Java platform to ensure high portability across our customers' platforms.

5. *Current Status*

Up to now important parts of the above-described SEEE have been implemented and validated. Main parts of it are shown in Fig. 12. This subpart of the SEEE is called intelligent retrieval and storage system (INTERESTS). In addition, some of the tasks shown in Fig. 2 have been validated using other technologies.

The first instantiation of INTERESTS was the CBR-PEB system (Althoff, Nick and Tautz, 1999). Its objective is to provide a repository of CBR systems (i.e., tools and applications). It is accessible over the web[3] since July 1998 and as of September 2000 it includes 60 (characterizations of) CBR systems. Besides contributing to the validation of INTERESTS, CBR-PEB was the first application where the Goal-Question-Metric technique was used for evaluating EBs (Nick et al., 1999).

The intelligent process and quality management (IPQM) system was the second instantiation of INTERESTS. Its objective is to provide—in collaboration with Fraunhofer Institute for Manufacturing

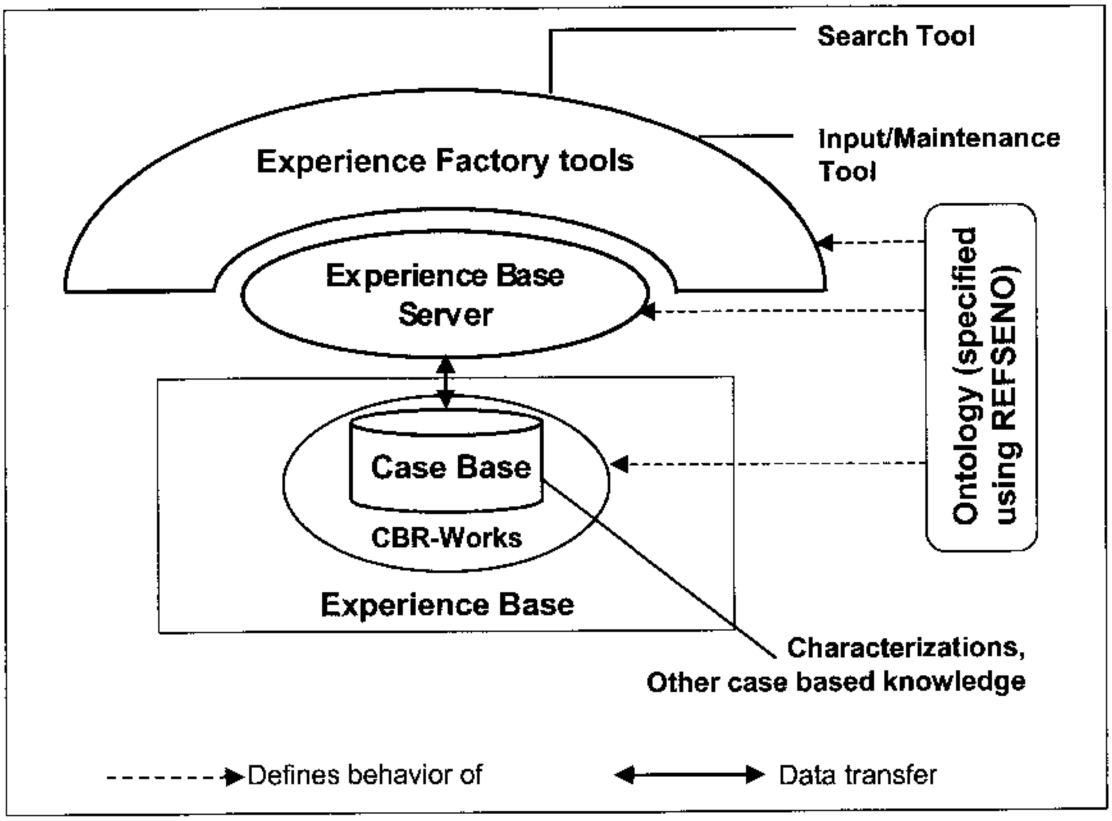


Fig. 12. Intelligent retrieval and storage system (INTERESTS).

Engineering and Automation (IPA)—a technical infrastructure for supporting continuous improvement processes in hospitals (Althoff et al., 1999). As of September 1999 it successfully passed field tests in three different hospitals and a demonstrator is publicly accessible over the web.[4] Besides validating INTERESTS and REFSENO technically, the IPQM project showed that both are also applicable for non-SE domains (see Fig. 13).

A third instantiation of INTERESTS is the knowledge management product experience base (KM-PEB). Its objective is to provide (a) a descriptive framework for knowledge management tools and approaches as well as (b) a repository of these tools that is accessible over the web.[5] Besides contributing to the validation of INTERESTS and REFSENO, KM-PEB also provided much experience on developing characterization schemas and instantiating them (see Fig. 14). In addition, it will be used for a roll-out of the evaluation program originally set up for CBR-PEB (Nick et al., 1999).

While the above three projects resulted in public demonstrator systems, there have been additional collaborations with industrial partners, which provided us with feedback for INTERESTS and the tasks presented in Fig. 2. An ongoing collaboration with Allianz Life insurance is concerned with the development of a lessons learned repository. Besides INTERESTS, we validated parts of our conceptualization that had been reused within this project. In a collaboration with DaimlerChrysler the goal was to build up experience about software inspections. Within this project we validated the recording tasks shown in Fig. 2.

The experiences gained in these and other projects resulted in a detailed description of the EF tasks (Tautz, 2000; Althoff et al., 2000) and a structured set of solution methods for these tasks. For this purpose research results from case-based reasoning, organizational memories, knowledge management, domain analysis, and software reuse have been combined (Tautz, 2000). The objective of this task-decomposi-

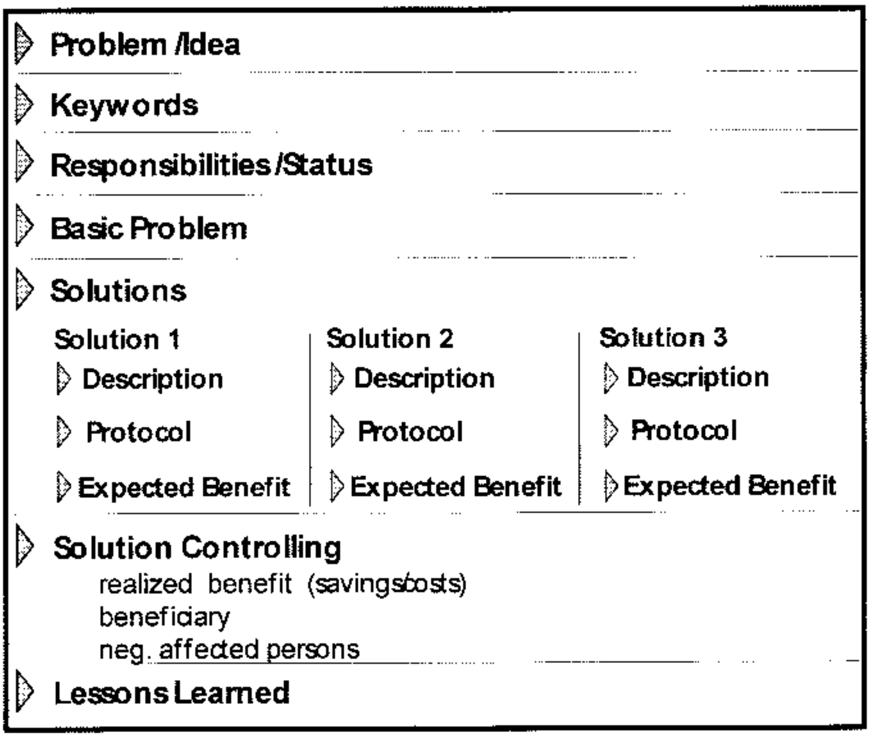


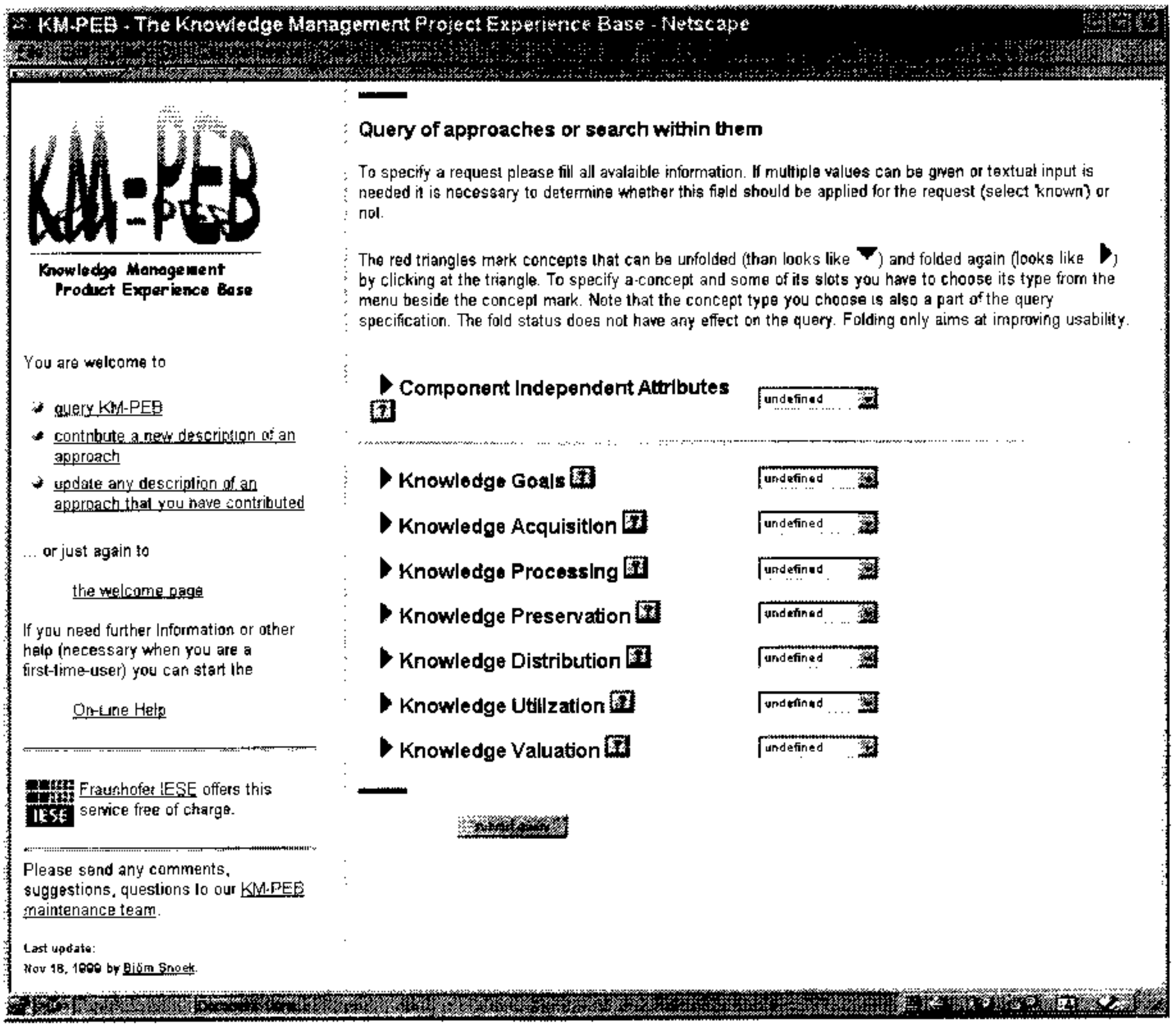
Fig. 13. *Main concepts in the IPQM system.*

tion hierarchy is to guide the tailoring to company-specific needs, to analyze employed processes for strengths and weaknesses, and to build up libraries for reusable problem-solving methods. The validation of these objectives is a major challenge and currently ongoing within the scope of building IESE's own EF (Tautz, 2000).

A first application tool (see Fig. 11) that currently is being technically integrated with INTERESTS is a tool for supporting the selection of SE technologies called KONTEXT (KnOwledge maNagement based on the application conTEXt of software engineering Technologies; see Birk and Kröschel, 1999; Birk, 2000).

6. *Discussion and Outlook*

The advantages and disadvantages of approaches like the EF/EB approach (see also Bergmann et al., 1999 and Bartlmae, 1999) with respect to knowledge maintenance and their comparison with approaches using more formal ontologies are currently discussed in the literature (see e.g., Menzies, 1999; Liao et al., 1999; Kalfoglou, 1999; Althoff, 1999a). Maurer and Holz (1999) argued that the processes involved in learning software organizations are very important (which is supported by Althoff et al., 2000) and showed how to integrate repository-based approaches with process-oriented ones. Snoek (1999) developed a descriptive framework for knowledge management approaches that explicitly considers this aspect among others (see also the description on KM-PEB). Weibelzahl (1999) describes a two-step CBR approach that is well suited to support processes for sales support in the web. Data mining approaches for automatically supporting the manual techniques for populating OMs are described in Anand, Aamodt, and Aha (1999). The evaluation of OMs as well as the particular knowledge assets within an OM are, for instance, discussed in Menzies and van Harmelen



KM-PEB

Knowledge Management
Product Experience Base

You are welcome to

- query KM-PEB
- contribute a new description of an approach
- update any description of an approach that you have contributed

... or just again to

the welcome page

If you need further Information or other help (necessary when you are a first-time-user) you can start the

On-Line Help

Fraunhofer IESE offers this service free of charge.

Please send any comments, suggestions, questions to our KM-PEB maintenance team.

Last update:
Nov 18, 1999 by Björn Snoek.

Query of approaches or search within them

To specify a request please fill all avalaible information. If multiple values can be given or textual input is needed it is necessary to determine whether this field should be applied for the request (select 'known') or not.

The red triangles mark concepts that can be unfolded (than looks like ▼) and folded again (looks like ▶) by clicking at the triangle. To specify a concept and some of its slots you have to choose its type from the menu beside the concept mark. Note that the concept type you choose is also a part of the query specification. The fold status does not have any effect on the query. Folding only aims at improving usability.

▶ **Component Independent Attributes**	undefined
▶ **Knowledge Goals**	undefined
▶ **Knowledge Acquisition**	undefined
▶ **Knowledge Processing**	undefined
▶ **Knowledge Preservation**	undefined
▶ **Knowledge Distribution**	undefined
▶ **Knowledge Utilization**	undefined
▶ **Knowledge Valuation**	undefined

Fig. 14. *Main concepts in the KM-PEB system.*

(1999), Bartsch-Spörl (1999), and Nick et al., (1999). A product experience base for machine learning algorithms is described in Lindner and Studer (1999).

Notes

1. The division into three levels is state-of-the-art, though they are called differently for different approaches (e.g., ''meta model'', ''class diagram'', ''instances'' in UML (RATIONAL (1997)), or ''modeling formalism'', ''data model'', ''data'' in the database community).
2. Statistical data can also be kept with the cases, so as to assess their usefulness (e.g., data on how often they had been applied successfully, or to what extent they had to be modified to be applicable).
3. http://demolab.iese.fhg.de:8080/
4. http://demolab.iese.fhg.de:8080/Project-KVP-EB/
5. http://demolab.iese.fhg.de:8080/KM-PEB

References

Aamodt A, Plaza E. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AICOM* 1994;7(1):39–59.

Althoff K-D. Validating case-based reasoning systems. *Proc. 5. Leipzi-ger Informatik-Tage*. Forschungsinstitut für Informationstechnologien e.V., Leipzig. 1997:157–168.

Althoff K-D. Panel on knowledge maintenance: Does meta-knowledge complicate KM? The CBR perspective. In *Proc. of the 11th International Conference on Software Engineering and Knowledge Engineering (SEKE'99)*. June 16 to 19, 1999:405.

Althoff K-D. *Evaluating Case-Based Reasoning Systems*. Springer Verlag, LNAI series, to appear, 2000.

Althoff K-D, Bartsch-Spörl B. Decision support for case-based applications. *Wirtschaftsinformatik* 1996;38(1): 8–16.

Althoff K-D, Birk A, Gresse von wangenheim C, Tautz C. Case-based reasoning for experimental software engineering. In: Lenz M, et al. eds. *Case-Based Reasoning Technology—From Foundations to Applications*. Springer-Verlag, 1998.

Althoff K-D, Birk A, Hartkopf S, Müller W, Nick M, Surmann D, Tautz C. Populating, utilizing, and maintaining a software engineering experience base. In: Bomarius F, Ruhe G, eds. *Learning Software Organizations-Methodology and Applications*. Springer Verlag, to appear, 2000.

Althoff K-D, Bomarius F, Müller W, Nick M. Using case-based reasoning for supporting continuous improvement processes. In: Perner P, ed., *Maschinelles Lernen—FGML'99—Proc. German Workshop on Machine Learning*. IBal Report, Institute for Image Processing and Applied Informatics, Leipzig, ISSN 1431–2360. 1999:54–61.

Althoff K-D, Nick M, Tautz C. CBR-PEB: A tool for implementing reuse concepts of the experience factory for CBR-system development. In Melis E., *Proc. 5th German Conference on Knowledge-Based Systems (XPS'99) Workshop on Case-Based Reasoning (GWCBR'99)*. also: IESE-Report No. 058.98/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany. CBR-PEB is publicly accessible via http://demolab.iese.fhg.de:8080/, 1999.

Althoff K-D, Wilke W. Potential uses of case-based reasoning in experience based construction of software systems and business process support. In: Bergmann R, Wilke W., eds., *Proceedings of the 5th German Workshop on Case-Based Reasoning*. LSA-97-01E, Center for Learning Systems and Applications, University of Kaiserslautern, 1997:31–38.

Anand SS, Aamodt A, Aha DW, eds. *Automating the Construction of Case Based Reasoners*, Proc. Workshop ML-5 at the 16th International Conference on Artificial Intelligence (IJCAI'99), July 31–August 6, 1999.

Bartlmae K. An experience factory approach for data mining. In: *Proc. 2nd Workshop "Data Mining and Data Warehousing as foundation of recent decision support systems" (DMDW99)*. LWA99 Collection, ISBN 3-929757-26-5, University of Magdeburg, September 1999:5–14.

Bartsch-Spörl B. Cases as knowledge assets. In: Schmitt S, Vollrath I., eds., *Challenges for Case-Based Reasoning—Proc. of the Workshops of the 3rd International Conference on Case-Based Reasoning*. LSA-99-03E, Center for Learning Systems and Applications, University of Kaiserslautern, 1999:I-25–I-28.

Basili VR, Caldiera G, Rombach HD. Experience factory. *Encyclopedia of Software Engineering*, volume 1. John Wiley & Sons, 1994:469–476.

Basili VR, Rombach HD. Support for comprehensive reuse. *IEE Software Engineering Journal* 1991;6(5):303–316.

Bergmann R, Breen S, Göker M, Manago M, Wess S. *Developing Industrial Case-Based Reasoning Applications—The Inreca Methodology*. Springer Verlag, 1999.

Biggerstaff T, Richter C. Reusability framework, assessment, and directions. *IEEE Software* 1987;4(2):41–49.

Birk A. *Knowledge Management of Experiences about the Application Context of Software Engineering Technologies*. PhD thesis, Department of Computer Science, University of Kaiserslautern, 2000.

Birk A, KRÖSCHEL F. A knowledge management lifecycle for experience packages on software engineering technologies. In Bomarius F, ed. *Proc. Workshop on Learning Software Organizations*. Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany. 1999:115–126.

CBR-WORKS (http://www.tecinno.com) 2000.

Feldmann R, Tautz C. Improving best practices through explicit documentation of experiences about software engineering technologies. *Proceedings of the International Conference on Software Process Improvement in Research and Education (INSPIRE '98)*. London, England, 1998.

Gresse von wangenheim C, Althoff K-D, Barcia RM. Intelligent Retrieval of Software Engineering Experienceware. In: *Proc. of the 11th International Conference on Software Engineering and Knowledge Engineering (SEKE'99)*. June 16 to 19, 1999:128–135.

Henninger S. Supporting the construction and evolution of component repositories. *Proceedings of the 18th International Conference on Software Engineering (ICSE'96)*. Berlin, Germany, 1996.

Henninger S. Capturing and formalizing best practices in a software development organization. *Proceedings of the 9th International Conference on Software Engineering and Knowledge Engineering (SEKE'97)*. Madrid, Spain, 1997.

Kleiner A, Roth G. How to make experience your company's best teacher. *Harvard Business Review* 1997;75(5):172–177.

Kalfoglou Y. Panel discussion notes: knowledge maintenance—The role of formal ontologies. *Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering (SEKE'99)*. Kaiserslautern, Germany, 1999:401–404.

Kotter JP. *Leading Change*. Harvard Business School Press, Boston, 1996.

Landes D, Schneider K, Houdek F. Organizational learning and experience documentation in industrial software projects. In: Decker S, Maurer F, eds., *Proceedings of the Interdisciplinary Workshop on Building, Maintaining, and Using Organizational Memories (OM-98)*. Brighton, UK, 1998:47–64.

Liao M, Abecker A, Bernardi A, Hinkelmann K, Sintek M. Ontologies for Knowledge Retrieval in Organizational Memories. In Bomarius F, ed. *Proc. Workshop on Learning Software Organizations*. Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, 1999:11–25.

Lindner G, Studer R. AST: Support for algorithm selection with a CBR approach. In: Perner P, ed. *Maschinelles Lernen—FGML'99—Proc. German Workshop on Machine Learning*. IBal Report, Institute for Image Processing and Applied Informatics, Leipzig, ISSN 1431–2360. 1999:62–71.

Maurer F, Holz H. Process-oriented knowledge management for learning software organizations. In *Proc. Twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW'99)*. Banff, Oct. 1999:16–21.

Menzies T. Knowledge maintenance heresies: Meta-knowledge complicates KM. In: *Proc. of the 11th International Conference on Software Engineering and Knowledge Engineering (SEKE'99)*. June 16 to 19, 1999:396–400.

Menzies T, Van Harmelen F. The second banff KAW'99 track on evaluation of KE methods. http://www.csc.wvu.edu/timm/banff99/. 1999.

Nick M, Althoff K-D, Tautz C. Facilitating the practical evaluation of organizational memories using the goal-question-metric technique. In: *Proc. Twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW'99)*. Banff. October 1999:16–21.

Ostertag E, Hendler J, Prieto-Díaz R, Braun C. Computing similarity in a reuse library system: An AI-based approach. *ACM Transactions on Software Engineering and Methodology* 1992;1(3):205–228.

Prieto-Díaz R. Implementing faceted classification for software reuse. *Communications of the ACM* 1991;34(5):89–97.

RATIONAL *Unified Modeling Language Version 1.1*. Rational Software Corporation, 1997.

Senge PM. *The fifth discipline: The art and practice of the learning organization*. New York. Doubleday Currency, 1990.

Snoek B. *Knowledge Management and Organizational Learning—Systematic Development of an Experience Base on Approaches and Technologies*. Diploma thesis, Department of Computer Science/Fraunhofer Institute for Experimental Software Engineering, University of Kaiserslautern. Available at: http://demolab.iese.fhg.de:8080/KM-PEB/. 1999.

Sumner T, Domingue J, Zdrahal Z, Hatala M, Millican A, Murray J, Hinkelmann K, Bernardi A, Wess S, Traphöner R. Enriching representations of work to support organisational learning. *Proceedings of the Interdisciplinary Workshop on Building, Maintaining and Using Organizational Memories (OM-98)*, Brighton, UK, 1998:109–128.

Tautz C. *Customizing Software Engineering Experience Management Systems to Organizational Needs*. PhD thesis, Department of Computer Science, University of Kaiserslautern, 2000.

Tautz C, Althoff K-D. Using case-based reasoning for reusing software knowledge. In: Leake D, Plaza E, eds. *Case-Based Reasoning Research and Development– Proceedings of the 2nd International Conference on Case-Based Reasoning*, Providence, RI, July 1997, Springer-Verlag, 1997:156–165.

Tautz C, Gresse von wangenheim C. REFSENO: A representation formalism for software engineering ontologies. *Proc. 5th German Conference on Knowledge-Based Systems (XPS99) Workshop on Knowledge Management, Organizational Memory, and Reuse.* Also: Technical Report IESE-Report No. 015.98/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern (Germany), 1999:61–71.

Uschold M, Gruninger M. Ontologies: Principles, methods, and applications. *The Knowledge Engineering Review* 1996;11(2):93–136.

Weibelzahl S. Conception, implementation, and evaluation of a case-based system for sales support in the Internet. Diploma thesis. University of Trier; Available at: http://www.cs.uni-sb.de/users/jameson/pdf/weibelzahl.pdf, 1999.

Klaus-Dieter Althoff graduated from Aachen University of Technology in Mathematics and Operations Research with an emphasis on Expert Systems in 1986. He received his Ph.D. in computer science on an integrated learning and problem solving workbench for technical diagnosis in 1992 and his Habilitation (post-doctoral degree) in computer science on the evaluation of case-based reasoning (CBR) systems in 1997. He is first author of the most detailed review of industrial CBR tools. He participated in a number of projects on knowledge-based systems, machine learning, and CBR at the University of Kaiserslautern (e.g., project leader for the Inreca Esprit 3 project), and on knowledge management, organizational learning, experience factory, and CBR at the Fraunhofer Institute for Experimental Software Engineering (IESE) (e.g., project leader for the build-up of IESE's experience factory). He was program chair of the 3rd International conference on CBR (July 27–30, 1999) and was/is involved in a number of German/international events on CBR, knowledge capture and management, and organizational and machine learning as a co-chair/program committee member/co-organizer. He is co-editor of the German journal on Artificial Intelligence and co-speaker of special interest group on machine learning of the German Computer Society. Since 1997 he leads the Experience Factory Technology group at IESE.

Frank Bomarius graduated from the University of Kaiserslautern in Computer Science and Electrical Engineering in 1986 and he received his Ph.D. in Computer Science in 1991. He then worked in an ESPRIT 2 project at the German Research Center for Artificial Intelligence in the area of Multi-Agent Systems. In 1993 he became a team leader and software developer at Tecmath GmbH. Since 1996 he is department head and since 2000 deputy director of the Fraunhofer IESE. Since 1996 he is transferring continuous, goal-oriented software process improvement (SPI) programs to software organizations in different industrial sectors. He applies the Quality Improvement Paradigm (QIP), the Goal/Question/Metric (GQM) and the Experience Factory (EF) approach. He is doing applied research and technology transfer in the area of Learning Software Organizations (LSO) and introduces EF-based knowledge management into industrial software engineering settings. His major focus is on the successful integration of knowledge management with existing organizational structures and work processes, alignment with ongoing improvement programs as well as the technical integration of experience bases with an organization's infrastructure. Dr. Bomarius has given numerous presentations at industrial as well as scientific workshops and seminars. He has 10 years of industrial experience and 15 years of experience in teaching and training of students as well as professionals. Since about 18 years he participated in European projects, namely Esprit 1 project PCTE (#8462), ESPRIT 2 project IMAGINE (#5362), and ESPRIT 4 project PROFES (#23239). Dr. Bomarius is serving as organizer, program chair and program committee member in national and international conferences and workshops in the area of software process improvement and knowledge management, such as SEKE, PROFES, CONQUEST, ICCBR, and LSO. Frank Bomarius is member of the IEEE Computer Society, the German Computer Society (GI) and the Working Group for Artificial Intelligence of the GI.

Carsten Tautz is a scientist at the Fraunhofer institute for Experimental Software Engineering. His research interests include the set-up of experience factories in general and the structuring of experience bases in particular. He recently completed his Ph.D. on Customizing Software Engineering Experience Management Systems to Organizational Needs. After receiving his diploma in computer science in 1993, he joined the Software Technology Transfer Initiative Kaiserslautern (STTI-KL). At the STTI-KL, Carsten Tautz was responsible for maintaining the technology package on "comprehensive reuse" and also took over some public relation functions regarding the transfer of innovative software engineering technologies. Currently, he is involved in an industrial cooperation concerning the introduction of a knowledge management program. In addition, he manages an international consortium with the aim of sharing software engineering experience among its members.