Organizational Learning and Experience Documentation in Industrial Software Projects

Dieter Landes, Kurt Schneider, Frank Houdek

Daimler-Benz AG
Research and Technology
P.O. Box 23 60
D-89013 Ulm, Germany
{landes, k.schneider, houdek}@dbag.ulm.daimlerbenz.com

Currently, several projects within our company focus on learning from experiences in the software domain. Yet, there are no textbook recipes how a process of organizational learning can be established. In particular, types of experiences must be identified that are potentially valuable for reuse. Furthermore, the organization and representation of such experiences must be defined in such a way that they can easily be retrieved and used for solving a new problem. In the paper, some mid-term insights are provided that were gained during the examination of these issues in one of the projects mentioned above.

1 Introduction

Learning from experience is getting more and more important in large companies. The demands of each company, and often in each business unit, however, may be very specific. We are currently involved in a large-scale initiative to foster learning from experience in the software domain.

There are numerous software development and software acquisition units in the Daimler-Benz Group. We were called in to establish within three of them a process of organizational learning (Houdek et. al. 1998). Learning should be tailored to the specific environment. As usual in industrial research, the predominant goal was not to obtain scientific results, but to produce benefits for the business units we were working with. Within two years the initiative was expected to result in a running experience exchange infrastructure, and a first indication of its usefulness.

Below, we use one of the business units as a real-world example to back our arguments. In that business unit, software quality management was the main topic to exchange experiences about. A number of very large administrative software projects needed support in planning, organization, and coordination of quality assurance activities. A systematic tracking of quality-related process artifacts was the goal. It was supposed to enable fine-grained estimation and control of quality aspects.

Basing an improvement initiative on experience has several advantages in such a domain. There are very little sources of knowledge available in literature or in company standards. In textbook-style publications, general principles are treated on an abstract level, but the essential details of how to apply those principles in a very specific situation are missing (e.g. Wallmüller 1995). It is the idiosyncrasies of a business unit that need particular attention to make quality management successful. And those particularities can never be reflected in a textbook.

Even company standards often cover too diverse environments to be of much use. They tend to be superficial and do in many cases not reflect company realities, according to their supposed users. Writing a project quality plan is one example where no theoretical treatment can compete with the experiences gained in that respective environment. Clearly, there can be generic templates for the structure of the document (such as, e.g., IEEE 1989) and a couple of generic guidelines (such as, e.g., IEEE 1995). These generic instructions, however, must still be tailored to reflect the specifics of the company in general and the project the document is intended for in particular. This tailoring depends mainly on experiences what worked well in similar projects in the given environment and what did not.

In this paper, we want to demonstrate how we classify, structure, and use experiences given the background sketched above. In Section 2, we give a brief overview of fundamental concepts of organizational learning as reflected in literature. The range of related work spans from organizational memories over experience factories to knowledge modeling. We confront the concepts with the goals and constraints we face in our working example. This discussion illustrates the tension between conceptual desires and pragmatic necessities. We are convinced that a similar situation exists throughout many industrial efforts of knowledge management. Our particular situation is described in Section 3.

In Section 4 we present how we respond to the tension. We describe the kinds of experiences we consider useful and how experiences can be classified. Closely related is the question how to represent and finally use experiences. We describe important points and trajectories we see in the spectrum from informal to formal representations. Several concepts and tools were developed to support those representations. Future directions are sketched in Section 5. Ontology annotation is a formal tool to support more powerful mechanisms of experience engineering. A static ontology, however, will probably fail to work in our environment. Therefore, we recommend a dynamic ontology that is able to coevolve with the domain, the maturity of captured processes, and our understanding thereof. Section 6 summarizes the core messages of the paper and discusses our main findings.

2 State-of-the-Art in Organizational Learning

Organizational learning has been a field of active study for quite a while (Senge 1990, Brown and Duguid 1991, Watkins and Marsick 1993). Basic principles common to most approaches are the capturing, storing, and reusing experiences or knowledge. Whereas approaches rooted in management science tend to emphasize the importance of system thinking (Senge 1990), computer scientists are more attracted to design rationale systems (Conklin and Begeman 1988; Moran and Carroll 1996) or organizational memories (Terveen et al. 1993).

Fischer et al. (1996) claim successful organizational learning requires a closed circle of knowledge (or experience) stimulation and externalization, maintaining the interest of the users, storing the surfaced information, and finally feeding it back when it is needed in the work context (Fischer 1994, see Figure 1).

In order to maintain user interest, each participant needs to feel a personal benefit. But also, the effort expended will determine whether the participant likes the system or process. In total, the *perceived utility* is the interesting concept:

$$\text{Perceived utility} = \text{benefit} / \text{expended effort}$$

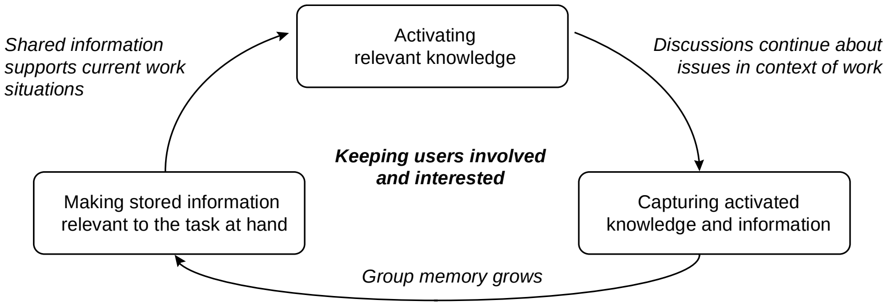


Fig. 1. A circle that needs to be closed at all times (Fischer et al. 1996)

According to this formula, one can target high benefit. This would justify some effort to be spent in return. gIBIS (Conklin and Begeman 1988) and – to a certain degree – answer garden (Ackermann 1994) follow this approach. Alternatively, one can avoid to promise much benefit (at least initially). In this case, effort needs to be kept low, too. There are examples for organizational memories (Lindstaedt 1996) and for design rationale systems that fall in this category (Schneider 1996).

However, it is an inherent property of experience systems to grow. This can lead from a low-effort/low-benefit initial phase to an increasing amount of both benefit and effort invested (Terveen et al. 1993, Ostwald 1995). We consider such a low-threshold/high-ceiling approach most promising.

In most of the work mentioned above, a specific tool is required. The tool is the cornerstone for the learning endeavor. The experience factory concept is very different in this respect, but not only in this respect (Basili et al. 1994a). An experience factory is an organizational unit rather than a tool. A group of people interacts with a software development unit and helps them to extract and reuse experiences (see Figure 2). Besides the roles of analyst and project supporter, there are so-called experience engineers who consolidate, compare, and add value to the original experience. There is an experience base to store experiences and related documents. However, how the experience base will be realized is specific to the environment where it is supposed to work: it can be just a shelf filled with folders, or it could be a sophisticated knowledge repository and reasoning system. This concept relies mainly on goal-driven measurement programs (Basili et al. 1994b) and focuses improvement that is supported by experience mechanisms. At NASA/SEL, the experience factory has been running successfully for many years (Basili et al. 1992).

Since organizational learning is based upon experiences and knowledge of people in a certain environment, it is evident that there is also a relationship between organization learning and work in the knowledge engineering area. In a certain sense, knowledge-based systems are also an attempt to store the knowledge and experiences of experts for particular tasks, such as, e.g., diagnosis, and make it available to a wider community of users. In contrast to organizational memories, however, knowledge-based systems are not primarily intended to give direct access to the knowledge they embody, but rather to allow this knowledge to be used for accomplishing a certain task even by less knowledgeable persons.

At the present stage of research, it does not seem possible to implement organizational memories in general by means of knowledge-based systems since knowledge-based systems presently work well only for limited tasks, such as diagnosis, and narrow domains, e.g., infectious diseases in medicine. In general, however, these preconditions are not met by organizational memories since there are usually broad domains (such as quality management in our case) and a diversity of tasks that are to be supported (such as, e.g., quality assurance planning, definition of quality goals etc).

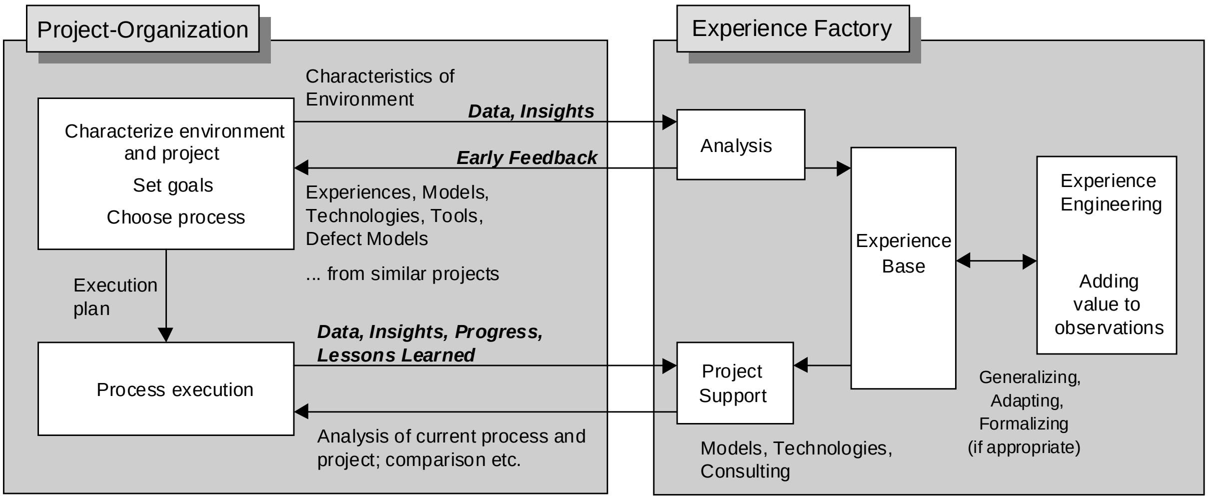


Fig. 2. Interaction of the experience factory with a software project organization

Nevertheless, knowledge engineering techniques are useful for capturing experiences (interview techniques, observation techniques etc.) and imposing structure on the captured information. In particular, ontologies are a promising means for the latter issue. Furthermore, ontologies and formal reasoning techniques might be useful for intelligent information retrieval mechanisms. We will come back to this issue in Section 5.

3 The Context of the Project and Some of its Implications

In this section, some background information concerning the particular project which will serve as our running example will be provided. In particular, some of the restrictions will be outlined that have important consequences on which types of experiences are available and how they are represented in the given environment.

In order to achieve the main project goal of enabling and supporting systematic learning from experience in the software domain, we decided to use the experience factory paradigm (Basili et al. 1994a) as the basis. This was mainly due to the fact that the experience factory paradigm is an extension of the QIP/GQM methods (Basili et al. 1994b) that we used already for quite a while in several process improvement projects. The experience factory was to be established on the basis of several application projects that are supposed to provide experiences, but which are also candidates for reusing experiences from other application projects. It is important to note that the primary goal of our project was not so much to extend and refine the theory behind the experience factory, but to produce benefits for the application projects by using the existing concepts the experience factory paradigm provides.

A couple of constraints turned out to be important:

- Within only two years the project is expected to establish a prototypical infrastructure for experience exchange and to provide first indications for the usefulness of the chosen approach.
- Within those two years, a basis for the continued operation of the experience factory has to be laid.

As the most important consequence of these constraints, it is essential to be able to demonstrate in early project phase that the experiences which the experience factory has to offer are really useful. This needs to be demonstrated to both the members of the application projects, and to the management. Therefore, it is important to capture as many relevant experiences as quickly as possible, no matter their format or representation media. This circumstance led to the development of the BIP paradigm which will be discussed in more detail below.

A major difference between our environment and the environment at NASA/SEL (Basili et al. 1992) is that at NASA experiences were primarily derived from quantitative data from measurement programs, whereas we had no such measurement programs in our application projects. Furthermore, it turned out that the application projects did not have sufficient access to experienced quality management experts. The project members who were in charge of quality management issues in our application projects were relatively new to the subject. As a consequence, they are very interested in getting support through experiences, but on the other hand they were initially not able to provide established best practices for further use in other projects. The kind of experiences they provided were basically reports dealing with the problems they encountered with particular issues, the reasons for these problems, the solutions they tried, and how well these solutions worked or why they (partially) failed, and so on.

In summary, from an experience factory perspective, this means that we cannot focus on the same types of information that are relevant in the NASA/SEL environment, but have to find out ourselves which types of information chunks are important in our specific context and through which levels of maturity these information chunks go during their lifecycle. The schema we developed will be presented below.

Already at an early stage of the project, the importance of browsing mechanisms became evident which allows potential users of experiences to scan at least parts of the material in the experience base. In general, users want to be able to determine on their own which information chunks in the experience base might be relevant to their current problem. As already discussed elsewhere (Landes and Schneider 1997), this has the consequence that a format for the contents of the experience base is chosen which allows a quick overview, and that the used notation is understandable to a wide range of people. On the other hand, representing the information in an informal manner only is insufficient since the contents of the experience base must be structured in some way for easy search and access. Information retrieval mechanisms must be more sophisticated than simple keyword search. This means that a whole spectrum of notations is required each of which serves a specific purpose. The notations we use will be presented below.

4 Documenting and Using Experience

4.1 What are Experiences in our Environment?

4.1.1 Experience Types

One of the main challenges during the instantiation of the concept of a learning organization is the clarification of experience types useful to look on since they able us to learn something. Due to the different environments, we could not borrow much from other known experience factories that could help to answer this question for our specific environment. Therefore, we adopted a bottom-up approach, i.e. in a first step, we collected as much experience-related-material as possible. We estimated subjectively the degree of usefulness of a piece of information to people different than the information originator. This was the criterion to decide whether the chunk of information was considered worth storing.

Software quality management as the domain which we are tackling has many facets and is subject to continuous improvement and research. The members of our application projects who were in charge of quality management issues were, at that time, relatively new to the subject. As a consequence the material we got in the beginning primarily consisted of problem statements about what they were currently trying to solve, the potential solutions they were about to try, and reasons why, in their opinion, available standards were only a limited help. Later, we also obtained information about how well the tried solutions actually worked or why some of them (partially) failed.

This information was collected primarily during interviews which were documented in interview transcripts and later analyzed. Analysis results were again documented. Besides, in the interviews, people often referred to project documents which were used as base documents or which they produced as a result of their work.

On the basis of these different types of experience-related material we encountered, we derived a classification of the respective documents with respect to their maturity, i.e. their degree of context-specificity and the amount of analysis involved. Table 1 reflects our current understanding about which type of documents can be classified to which maturity level.

Maturity level:	Raw context-specific documents	Auxiliary documents	Experiences / Lessons learned
Document types:	• Contract • Project (quality) plans • Project task sheets	• Interview transcripts • Document indexes • Analysis notes	• Guidelines • Checklists • Process descriptions • Analysis summaries

Table 1. Maturity levels and document types

Source	**Stimulus**	**Collection**	**Validation**
Solicited qualitative observation	A problem notification, a GQM plan leading to an unclear issue, a side-remark in an interview	Specific focused interviews asking for this issue, supplemented by notes and conclusions afterwards	Additional interviews; check of documents mentioned or found relevant
(Improvement) rationale	Decision made or design produced for a particular task	Introspection; interviews	Critical check of whether it is rationale or a (post-mortem) rationalization
Problem observation	„Bad experience“ someone currently suffers from in a project	Remark of affected person; then general or focused interview	–
Success story	Announcement of a (unexpected or underestimated) success that goes beyond pure public relations	Interviews	Checks of documents and details that could support (or contradict) the announcement; Interviews with other involved parties and comparison of answers
Diary study	Obviously relevant task is accompanied by writing notes	Notes taken during the task, with goals and plan in backhead	–

Table 2. Experience-related material and associated activities

Furthermore, we generalized our observations into a kind of checklist which, among other things, indicates for several types of experience-related material the stimuli which cause these types of material to be produced, how they can be collected, analyzed, validated, etc. Table 2 shows a small part of this checklist.

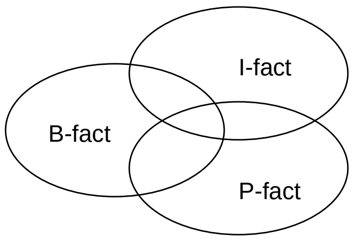
We found these classifications useful in two ways. First, they clarified our picture which pieces of experience-related material needed to be stored in the experience base (as they contain important information that is prone to be reused). Secondly, we gained deeper insights in which situations our project partners tended to raise important information that needs to be captured systematically.

4.1.2 Media

In the previous paragraphs, we focused on experience-related material that is present in written form. We found, however, that this is only one format in which such material may exist. For instance, similar to knowledge acquisition, much of the relevant knowledge or many relevant experiences are tacit, i.e. not available in explicit form right away. These implicit experiences, however, are also valuable for reuse in an experience factory context. Clearly, in the long run it is desirable to make these tacit experiences explicit since they can then be used without entirely depending on the individual that carries them in the back of her head. However, this elicitation is not possible without some sophistication. Simply asking is not enough.

Yet, it is impossible to capture all knowledge, at least for economical reasons. As a consequence, we have to accept that there will always a whole spectrum of documentation 'media': it ranges from undocumented knowledge in peoples' brains over raw and auxiliary paper documents (e.g. contracts from a particular project) to more structured objects (e.g. prescriptive process models of acceptance testing) which may exist in a more machine-readable format. As a model of this situation, we devised the so-called BIP paradigm: BIP is an acronym for *B*rain-*I*nternet-*P*aper, and the BIP-paradigm serves two functions (see Fig. 3), namely

- it documents our understanding that there is a whole range of valid formats for experience, and
- it suggests the appropriate medium to store a particular kind of knowledge.



B-fact:	Uncomplete, evolving things
I-fact:	Things which exist in many variants
P-fact:	Mainly external information (e.g. fax)
B-fact ∩ I-fact:	Important information build internal
I-fact ∩ P-fact:	(External) things which are useful also internally
P-fact ∩ B-fact:	Important knowledge from external
B-fact ∩ I-fact ∩ P-fact:	Information concerning the core of daily work

Fig. 3. The BIP paradigm

The BIP paradigm also documents our understanding that the experience factory is, even in the long run, not just a more or less intelligent experience repository, but that it will always contain a human component.

4.2 Describing and Using Experiences

In the previous section, we discussed different classification dimensions for classifying experience factory-related information. In the following sections, we are going to discuss still another aspect, namely the question what representation is appropriate for experience-related material. Since there is a spectrum of documents that contain relevant information, there will be no single formalism that is suitable for all of them. In the following, we will therefore point out what type of representation (informal, semi-formal, formal) seems to be appropriate for which purpose.

4.2.1 Informal Documentation

In essence, the BIP paradigm expresses the fact that every information may be valid as an experience, no matter what particular format of medium is used. As a consequence, the representation that is used for expressing these information must be seen as an acceptable one. Therefore, informal text, in particular, is a valid representation since it is used in documents at lower maturity levels, such as raw project documents.

Informal text has a couple of advantages. The most important strength is its suitability as a communication means since it is the representation that all potential recipients of experiences easily understand. Conversely, since no special training is required to get used to the representation, members of application projects can easily express their experiences on their own using that representation. The latter is also supported by the fact that informal text has unlimited expressive power – members of the project team can express what they want to express without struggling with the constraints imposed by a more formal and, thus, more restrictive representation.

Furthermore, informal text is also a 'cheap' representation format since all the relevant project documents already use this representation. This is particularly important in a situation like ours where a timely proof of concept for the experience factory approach is required (see Section 3) since no time-consuming translation to another representation is required before a piece of experience can be reused.

An informal representation alone, however, is not sufficient since relevant information may easily be hidden among a bunch of details which may be less important to the problem at hand. As a consequence, structure must be imposed in a suitable way to support the navigation through the existing material and the identification of the relevant pieces of information, yet without compromising the advantages of informal representations. To that end, we use semi-formal notations which are discussed in more detail below.

4.2.2 Semi-Formal Documentation

A more structured representation as a supplement to informal, mainly unstructured text provides a couple of additional benefits such as, e.g.,

- the descriptions of the experiences are easier to understand;
- a common structure establishes a common ground for experience interchange between different organizations; and
- tool support (e.g. search mechanisms) may use this structure as a basis.

For the definition of a suitable structure, we decided to shift the effort towards writing in order to simplify the reuse of described experience. Criteria for a suitable structure were

- the documented experience should be easily comprehensible, and
- the structure should not limit the ability to express things.

The only suitable approach which complies with both criteria is based upon structured text. Building on that premise, we developed *quality patterns* (Houdek and Kempter 1997; Houdek 1997) as a representation primitive to describe experiences in a structured manner.

4.2.2.1 Quality Patterns

A quality pattern consists of three main elements, namely a classification part, an experience part, and an explanation part (see Fig. 4).

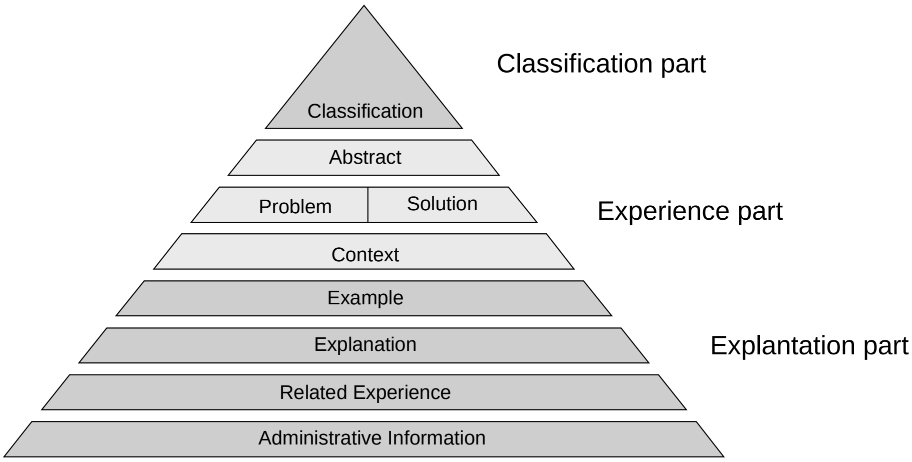


Fig. 4. Structure of a quality pattern

In the classification part, the described experience is classified with respect to a number of facets and additionally some keywords. The facets for classification emphasize the object, purpose, and intended group of readers of the described experience.

The experience part contains the main information. At this point, it is necessary to understand one of the basic concepts of quality patterns, i.e. the idea of patterns (Alexander 1979). According to this idea, the information is documented not in a result-oriented way which is normally used, but in a problem-oriented way. This makes it easier for a reader to retrieve the desired information since she typically is trying to solve a particular problem without knowing at that point what a suitable solution might be. Another essential element is the description of the context in which the experience was gained.

The cognitive model underlying patterns is based on the hypothesis that software development is (at least to a significant extent) deterministic. The application of the same technique, method, or process will result in (quite) the same result.

The third part of the quality pattern is for explanation purposes. An *example* illustrates the problem-solution pair. The *explanation* provides a justification for the proposed solution, e.g. by presenting data from a measurement program or supplying results from a real software project. The section *related experience* refers to other experience packages or related documents such as data collection forms or meeting notes. The *administrative information* mentions the author or the date of creation.

The second basic concept underlying quality patterns is the idea of pyramid thinking (Minto 1987). Information is presented in several layers. The top layer contains only the most important information, while the bottom layer provides all the details. Such an approach allows the reader to judge very early whether the presented experience is relevant for the required purpose or not. To make this concept more clear, we have depicted a (not completely filled in) quality pattern in Fig. 5.

Due to their underlying philosophy, quality patterns are primarily intended as a representation primitive to express types of experiences that are at a high maturity level, such as, e.g., guidelines.

4.2.2.2 Refinement of Quality Patterns

In the concept of quality patterns, the author of a piece of experience has to spend the main effort that is required for making it reusable. One major problem in writing a quality pattern is

Classification	Package type: Practice Object type: Product Object: IT-contract Viewpoint: Quality manager Environment: Large projects, out-sourcing of SW devel. Analysis technique: Observation Peculiarities: –
Abstract:	Four issues for reviewing and controlling the main and urgent issues in IT-contracts.
Problem:	Which issues should be considered in reviewing and correcting IT-contracts from the perspective of a quality manager.
Solution:	The following four issues can be checked in a contract even under time pressure: 1. Incorporate a clause like: “The acceptance process and –criteria still have to be regulated in mutual agreement. A relevant (written) agreement becomes a constituent of this contract.” 2. Incorporate performance criteria, either directly or by a clause like (1). 3. Take care of assumptions: Consider for each assumption like “the installation will take 30 minutes” what will happen if the assumption does not hold. 4. Check the assured resources on your side (e.g. for acceptance activities)
Context:	Large projects; software development is outsourced; the delivered software will be part of a larger system; quality management for the large system is made in-house
Example:	<not filled in here>
Explanation:	<not filled in here>
Related Exp.:	▪ Document “Example contract” ▪ Lessons learned package “Contract management in project XYZ”
Admin. Inf.	<not filled in here>

Fig. 5. Example of a quality pattern

the decision whether a particular situation (which lead to experience) or the gained experience (which is in fact more a subjective appraisal than a proven reality) should be described. In order to alleviate this problem, the concept of quality pattern is further refined into three subcategories, namely theory patterns, practice patterns, and lessons learned patterns (see Fig. 6).

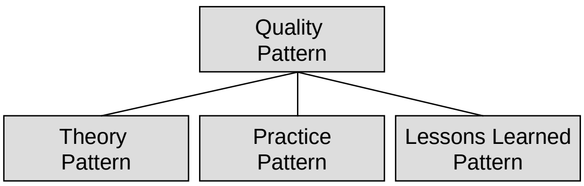


Fig. 6. Refinement of quality patterns

A theory pattern documents external knowledge such as, e.g., information from textbooks, conference proceedings, or seminars. Lessons learned patterns are used to describe an observation which could be drawn from a fact that really happened. As an option, it may contain a subjective judgement. A lessons learned pattern does not contain all parts of a normal quality pattern. The example and frequently also the explanation sections are missing. Practice patterns are used to package prescriptive things such as, e.g., a standard process. These three types of experience packages are useful to model the evolution of experience over time. Initially, there is some theory (e.g. adapted from textbooks). This theoretical knowledge can be used to build a tailored, prescriptive treatment. Applying the information captured in such a practice package will give rise to several experiences which will be packaged in one or more lesson learned packages. After some time of learning, a new version of a (prescriptive) practice package is created on the basis of the lessons learned packages together with the old practice package. Fig. 7 shows this evolutionary process graphically.

Our experience shows that writing a full-fledged quality pattern from scratch is a fairly difficult task, which is simplified considerably by following this evolutionary model.

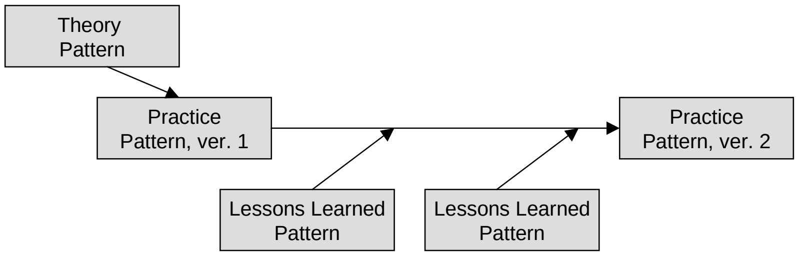


Fig. 7. Evolution of quality patterns

4.2.2.3 Tool Support

In order to make quality patterns usable, also a simple web-based tool was developed. Web technology was used since it might help to integrate the various system platforms that are used across a large enterprise such as ours. The main features of this tool are

- simple navigation through a system of quality patterns and auxiliary documents using hyperlinks, and
- searching by using facet classification or full-text retrieval.

More details about these tools can be found in (Fellger 1996) and (Bierer 1997).

4.2.2.4 Modeling the Relationships between Experience Documents

We observed that managing even a relatively small collection of only 10 or 20 experience packages becomes fairly complicated. This is not so much due to the content of the experience packages, but rather to their interrelation. Every new lessons learned or practice package relates to many other existing packages. Besides, many other (auxiliary) documents are also tied to the packages and may evolve.

To deal with this problem, we decided to capture the relationships between the different packages and documents more formally in a meta data model.

As a first task in building such a meta model, we had to analyze which types of entities were connected by which types of relationships in our particular context. Thereby we enlarged our scope from quality patterns to auxiliary documents. By this, new relationships beside the evolutionary relations (i.e. *a* is derived from *b*, and *a* is related to *b*) were identified. The results of this analysis are summarized in Table 3 and Table 4. In principle, all relationship types also have an inverse; due to limited space, only one direction is mentioned in the tables.

Entities are associated with a facet classification similar to the classification of quality pat-

Experience Packages	Actual experience packages, typically documented in quality patterns.
Context description	Description of the context. The rationale for extracting this information from the quality patterns is that typically a number of documents or experience packages are settled in one context. So extracting the context description helps to avoid redundancy.
Protocols/Notes	These documents are created during meetings or presentations. They help to document rationales or observations.
Raw data	Data, especially from measurement programs.
Auxiliary Documents	Documents which were used or produced within a software development project. In our case documents such as contracts, checklists or standards.

Table 3. Identified entities

terns. All entities can be linked with related documents or quality patterns, respectively. In order to avoid too much effort in building a consistent and complete model, the meta model can be dynamically extended with additional entity types or relationships (Rudolph 1997).

a is derived from *b*	Statement in document *a* is more concrete than *b*.
a includes *b*	Document *b* is a part of document *a*.
a describes *b*	This relationship is to express the relationship of a context description and another document.
a is related to *b*	This relationship is used to model the semantic evolution, e.g. the incorporation of lessons learned packaged in a new practice package.
a supports *b*	Relationship between documents of the same kind, e.g. experience packages, if their according statements support each other.
a contradicts *b*	Relationship between documents of the same kind, e.g. experience packages, if their according statements contradict each other.

Table 4. Identified relationships

Using this meta model as a conceptual basis conveys several benefits:

(1) the understanding of the documents related to one topic can be improved;

(2) some forms of consistency with respect to the used relationships can be checked automatically, and

(3) searching can be supported and focussed.

The first issue is primarily addressed in a tool (Rudolph 1997) which helps to manage the documents under the control of the experience factory and the relationships between them. Currently, the tool supports only the graphical management of the entities and relationships, but we plan to incorporate parsers and unparsers at least for quality patterns. Fig. 8 shows a screenshot of one of the tool's windows.

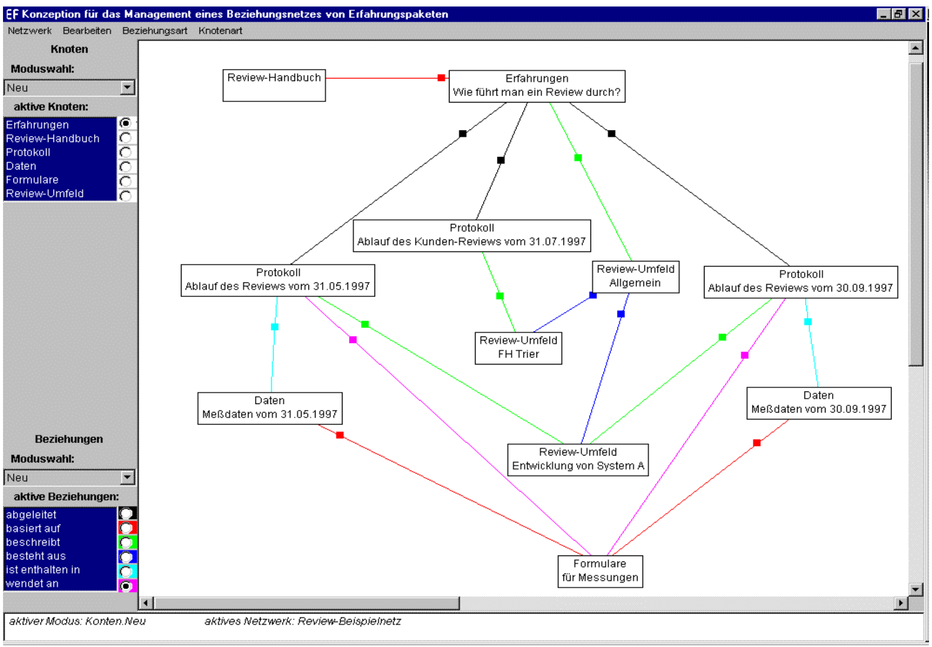


Fig. 8. Screenshot of the Experience Package Management Tool.

Consistency checking is related to the definition of the relationships. All types of relations are associated with rules which defines their compatibility with other relations. For instance, for no two packages *a* and *b*, *a* supports *b* and *a* contradicts *b* can be true at the same time (even over a couple of packages in between).

The support of searching tasks is achieved by distinguishing relationship types which allow inheritance (e.g. supports) from those which do not (e.g. contradicts). The search algorithm assumes that classification attributes are propagated through the network along relationships that allow inheritance.

We can already share some observations concerning these three features. Consistency checking and graphical manipulation proved to be much more useful. They helped to manage the complexity of the relationships between experience packages. The search facility turned out to be of less use since the mechanism of attribute inheritance is weak, especially because attributes are inherited syntactically without further semantic information.

4.2.3 Formal Documentation

We just pointed out that a search facility might benefit from including semantic information. One means to be able to include the semantic content of the documents in a more comprehensive way is the use of a formal representation formalism. Clearly, it is not feasible to formalize all documents completely since the effort required would be much too large. Furthermore, a formal representation is not suited for communication since the recipients of experiences are in general not trained to understand such representations. Therefore, a formal representation can only be useful as a supplement to the other notations described above.

Although we do not yet use formal representations so far, we intend to use formally described ontologies as a means to support experience classification and retrieval. This issue shall be elaborated in more detail in the next section.

5 Future Directions

5.1 Ontology Annotation

One of the most important results of knowledge engineering research during the last decade is the insight that ontologies play a fundamental role for the development of knowledge-based systems. Like problem-solving methods, ontologies are a component of a knowledge-based system that is amenable to be shared and reused (see, e.g., Pirlein and Studer 1995 or van Heijst et al. 1997). Ontologies are also a mechanism to guide the acquisition of knowledge by providing a modeling schema that indicates which specific types of knowledge are, e.g., relevant for a particular task. Furthermore, ontologies can be used to impose structure on a body of knowledge.

Currently, research focuses on the role that ontologies can play for intelligent information retrieval (Fensel et al. 1998). Specifically, the problem is tackled that the retrieval of relevant pages from the world-wide web is currently restricted to keyword-based search. Thus, neither the semantic content of the web pages is taken into account, nor can any information be retrieved which is not represented explicitly since no inferencing capabilities are supplied. The approach taken by Fensel et al. (1998) uses ontologies as a means to enrich documents in the world-wide web in order to represent semantic information. The ontology, in turn, is expressed in terms of a logical language which can be used by an inference engine to answer queries.

There is an evident analogy between the retrieval problem for world-wide web documents and for information from a large experience base. Although the size of our experience base is still relatively moderate, we are planning to use ontology annotation to support the identification of relevant information from our body of machine-readable experience documents. The first step in this undertaking will be an analysis of the experience documents that are currently available in order to build a suitable ontology. This ontology will cover aspects of the application domain as well as meta issues such as the relationships between different pieces of information. For the latter part, we will use our analysis of relationships between experience documents (see section 4.2.2.4) as a basis. Clearly, also work on design rationale (see, e.g., Moran and Carroll 1996) may provide important inputs. In the second step, this ontology will then be used to annotate the available experience documents. In the third step, a prototypical inference engine will be used for supporting the retrieval of relevant pieces of information.

As a by-product, we also expect the ontology to be a useful tool to impose structure onto the contents of the experience base. This is useful for retrieving information manually since it is relatively easy to find the experience documents that refer to a particular part of the ontology. Additionally, the ontology might also come in handy when additional information is inserted into the knowledge base since the new piece of information can more easily be linked to related pieces of information which refer to the same or closely related parts of the ontology.

5.2 Ontologies that Grow with You

As mentioned above, ontologies can enable numerous mechanisms for searching, modifying, and managing a base of experience-related documents. Furthermore, they can be a guidance for an experience package author by offering ordering and structuring frameworks. As our experiences have shown, however, neither our domain (software quality management), nor the – conceptually open – set of tasks the experience infrastructure is supposed to carry out is sufficiently clear-cut to tolerate a strict and static ontology.

Fischer and Nakakoji (1992), for example, argue against the claim of complete formal coverage in cases in which an open borderline call for creative interventions. They consider it simply inappropriate to spend a significant effort just to mimic intelligent tasks that can easily be carried out by humans. In many cases (ours is one) there are people available who can carry out experience engineering tasks. If ontology annotation is meant to be successful, we will have to apply a moderate approach characterized by the following statements.

- The ontology cannot cover all documents and all interesting aspects they may have. There will be – by definition – always a set of unregistered or only partially formalized documents.
- The ontology is relatively easy to modify (which is a non-trivial requirement, since each modification in the ontology can have ripple effect on many instances).
- The ontology is particularly easy to extend. When the experience author or experience engineer feels that the given ontology is insufficient to describe a given document or experience, she must face no high threshold to extend it.
- Following the 80-20 rule, the ontology and the mechanisms built on it cover effort-prone, repetitive tasks that can be easily automated. It does acknowledge the fact that it may not completely cover the set of experiences captured, and it makes this fact obvious to the users.

While all mechanisms provide support, the human is clearly the one in control. The human constantly has the option to contradict, to adapt, or to ignore advice from any mechanism (Fischer and Nakakoji 1992).

6 Discussion

In the initial phase of an organizational learning initiative, many issues have to be clarified. One of the most important ones is the question what is valid as an experience in a particular context. Since this cannot be answered in general and since there were no contexts that were sufficiently similar to ours, we took a bottom-up approach to collect relevant pieces of information, regardless of their format. The advantage of such an approach lies in the fact that it quickly leads to an initial population of potentially reusable experiences. Therefore, it is easier to achieve „early wins“ that convince people of the benefits of the experience factory approach, thus making it easier to motivate additional people to contribute their experiences.

The down-side of a bottom-up approach, however, is the lack of structure in the information. In order to cope with this problem, we analyzed an initial set of experience-related material to identify which types of information items can be distinguished and how they can be structured suitably. Classification dimensions that are useful for structuring the material we found are input source and form of analysis, but also maturity. Using these classification schemes, we were able to decide which types of experiences items are the ones that should be presented to a potential user first in order to allow him a quick judgement of how relevant the items are for the problem at hand.

Clearly, the question which types of experience related material are the relevant ones cannot be answered in general, but only relative to a particular context. Therefore, the experience types we identified can only be a starting point to answer this question for a different setting. We feel, however, that our approach to characterize which types of interesting material exist in a particular situation is transferable to other domains even outside the quality management or software domain.

A quick evaluation of candidate experiences must also be supported by appropriate notations. To that end, quality patterns are used as a semi-formal representation primitive which builds on principles such as pyramid thinking. Such a notation is useful for a more focused access to experience-related material than could be provided with informal text alone. It is important to note, however, that quality patterns are a representation primitive that will only be used for specific forms of experience-related material. Other (less mature) forms will always be described only informally. In addition to informal and semi-formal notations, we currently intend to annotate pieces of information with formally described ontologies in order to provide more sophisticated retrieval facilities.

Of course, one of the crucial issues in an effort to learn from experiences is the retrieval of information that is relevant to a problem at hand. One of the key messages of the experience factory paradigm is that this will never be accomplished in a completely automated fashion. In the current stage of our project, retrieval is accomplished manually by the project support team, i.e. so far, there is no direct access to the experience base for members of the application projects. Manual retrieval means that the project support team has a good knowledge of what the contents of the experience base currently are and which of the pieces might be appropriate to solve a problem at hand. Such an approach is still viable due to the currently limited size of the experience base (roughly 100 different documents of various types). Also, much of the reused material has been transferred to their recipients in verbal form, supplemented by some of the material in written form as it is described in the experience base. Although the direct personal interaction between project support team and application projects seems to be one of the crucial factors for successful reuse of experiences. More sophisticated aids for identifying potentially reusable pieces of information are needed if the experience base grows larger. Currently, the available retrieval mechanisms are fairly limited: there are some key-word-based search engines, and quality patterns by their very structure facilitate a quick overview if they contain information that is relevant to a particular problem. Yet, even if there were no direct access to the experience base for the members of the application projects, at least the project support team needs to have appropriate retrieval mechanisms. Due to limited resources, however, we cannot tackle the retrieval issue in sufficient depth in our current project, but it will be one of the main topics of a follow-up project.

Summarizing, we can state that although the overall feedback concerning our approach was positive, further evaluations in the real-life context of additional application projects must be performed. Since we have only initial answers to quite a few of the important issues in an organizational learning effort, we are sure that time will bring further refinements of the concepts we developed so far.

7 References

Ackermann, M.S. (1994): Augmenting the organizational memory: A field study of Answer Garden. In Proc. Conference on Computer Supported Collaborative Work (CSCW´94), Chapel Hill.

Alexander, C. (1979): *The Timeless Way of Building*. Oxford University Press, Oxford.

Basili, V.; Caldiera, G.; McGarry, F.; Pajersky, R.; Page, G.; Waligora, S. (1992): The software engineering laboratory – an operational software experience factory. In Proc. 14th International Conference on Software Engineering (ICSE), pp. 370-381.

Basili, V.; Caldiera, G.; Rombach H.D. (1994a): Experience factory. In: Marciniak, J. (ed.): *Encyclopedia of Software Engineering, vol. 1*. Wiley, New York , pp. 469-476.

Basili, V.; Caldiera, G; Rombach H.D. (1994b): Goal question metric paradigm. In: Marciniak, J. (ed.): *Encyclopedia of Software Engineering, vol. 1*. Wiley, New York, pp. 528-532.

Bierer, W. (1997): A procedure to create quality patterns. Diploma thesis, University of Stuttgart (in German).

Breuker, J.; Van de Velde, W. (eds.) (1994): *The CommonKADS Library for Expertise Modelling*. IOS Press, Amsterdam.

Brown, J. S.; Duguid, P. (1991): Organizational learning and communities-of-practice: Toward a unified view of working, learning, and innovation. In *Organization Science 2(1)*, pp. 40-57.

Conklin, P.; Begeman, M. (1988): gIBIS: A hypertext tool for exploratory policy discussion. In *Transactions of Office Information Systems 6(4)*, pp. 303-331.

Fellger, T. (1996): Structuring and classifying experience with hypertext-based realization. Diploma thesis, University of Mannheim (in German).

Fensel, D.; Decker, S.; Erdmann, M.; Studer, R. (1998): Ontobroker or How to enable intelligent access to the WWW. In Proc. 11th Knowledge Acquisition for Knowledge-Based Systems Workshop KAW'98, Banff, Canada.

Fischer, G. (1994): Turning breakdowns into opportunities for creativity. In: Special Issue on Creativity and Cognition, *Knowledge-Based Systems 7(4)*, pp. 221-232.

Fischer, G.; Lindstaedt, S.; Ostwald, J.; Schneider, K.; Smith, J. (1996): Informing system design through organizational learning. In Proc. International Conference on Learning Sciences (ICLS'96), pp. 52-59.

Fischer, G.; Nakakoji, K. (1992): Beyond the macho approach of artificial intelligence: Empower human designers – do not replace them. In *Knowledge-Based Systems 5*, pp. 15-30.

Gruber, T.R. (1993): A translation approach to portable ontology specifications. In *Knowledge Acquisition 5(2)*, pp. 199-220.

Houdek, F. (1997): Software quality improvement by using an experience factory. In Dumke, R.; Lehner F.; Abran A. (eds.): *Software Metrics – Research and Practice in Software Measurement*. Deutscher Universitätsverlag, pp. 167-182.

Houdek, F.; Kempter, H. (1997): Quality patterns – An approach to packaging software engineering experience. In: Harandi, M. (ed.): Proceedings of the Symposium of Software Reusability (SSR'97), *Software Engineering Notes 22(3)*, pp. 81-88.

Houdek, F.; Schneider, K.; Wieser, E. (1998): Establishing Experience Factories at Daimler-Benz: An Experience Report. In Proc. 20th International Conference on Software Engineering (ICSE), pp. 443-447.

IEEE Standard for Software Quality Assurance Plans (1989). IEEE-Std. 730-1989, Institute of Electrical and Electronics Engineers, New York.

IEEE Guide for Software Quality Assurance Planning (1995). IEEE-Std. 730.1-1995, Institute of Electrical and Electronics Engineers, New York.

Landes, D.; Schneider, K. (1997): Systematic editing and using experience from software projects at Daimler-Benz. In: Oberweis, A.; Sneed, H.M. (eds.): *Software Management '97*. Teubner, Stuttgart and Leipzig, pp. 63-73 (in German).

Lindstaedt, S. (1996): Towards organizational learning: Growing group memories in the workplace. CHI doctoral consortium.

Minto, B. (1987): *The Pyramid Principle - Logic in Writing and Thinking*. Minto International, London, 3rd edition.

Moran, T.; Carroll, J. (1996): *Design Rationale: Concepts, Techniques, and Use*. Erlbaum, Mahwah, NJ.

Newell, A. (1982): The knowledge level. In *Artificial Intelligence 18*, pp. 87-127.

Ostwald, J. (1995): The evolving artifact approach: Knowledge construction in collaborative software development. Ph.D. Dissertation, University of Colorado, Boulder.

Pirlein, T.; Studer, R. (1995): An environment for reusing ontologies within a knowledge engineering approach. In *International Journal of Human Computer Studies 43(5/6)*, pp. 945-965.

Puppe, F. (1993*): Systematic Introduction to Expert Systems: Knowledge Representation and Problem-Solving Methods*. Springer, Heidelberg.

Rudolph, A. (1997): Conception for the management of relationships of experience packages. Diploma thesis. Trier Polytechnic.

Schneider, K. (1996): Prototypes as assets, not toys. Why and how to extract knowledge from prototypes. In Proc. 18th International Conference on Software Engineering (ICSE), Berlin.

Senge, P. (1990): *The Fifth Discipline - The Art & Practice of the Learning Organization*. Random House, London.

Terveen, L.G.; Selfridge, P.G.; Long, M.D. (1993): From folklore to living design memory. Human factors in computing systems. In Proc. INTERCHI´93, pp. 15-22.

Van Heijst, G.; Schreiber, A.T.; Wielinga, B.J. (1997): Using explicit ontologies in knowledge-based system development. In International Journal of Human-Computer Interaction.

Wallmüller E. (1995): Holistic quality management in information technology. Hanser (in German).

Watkins, K.E.; Marsick, V.J. (1993*): Sculpting the Learning Organization – Lessons in the Art and Science of Systematic Change*. Jossey-Bass, San Francisco.