IESE

Fraunhofer Einrichtung Experimentelles Software Engineering

Transferring Experience: A Practical Approach and its Application on Software Inspections

Authors:
Frank Houdek
Christian Bunse

Submitted to
Workshop LSO at SEKE'99

Partially supported by
BMBF-Förderung Software Technology
FKZ 01 IS 518

IESE-Report No. 008.99/E
Version 1.0
February 1999

A publication by Fraunhofer IESE

Abstract

Experience and knowledge management are seen as key capabilities for systematic software development and process improvement. However, it is still not quite clear, how to get this vision to work.

In this paper, a process for systematic experience transfer is presented. It covers the activities of experience acquisition, experience documentation and evolution, and experience reuse. This process is a result of the German publicly-funded project SoftQuali, and its practical use is demonstrated by two real project examples, dealing with experience transfer for software inspections. In general it is described how experience can be packaged, both to transfer the technique and to improve it.

Keywords: Experience transfer, inspections, quality pattern, organizational learning, experience management.

Table of Contents

1 Introduction

Software is becoming more and more important. This importance leads to an increase in the demand for high quality software products. Software as well as its development processes have to be improved continuously. To do so, best practices have to be identified, maintained, and transferred systematically from one project to another. A continuous, company-wide learning cycle must be established to transfer knowledge, avoid mistakes, and thereby improve both processes and products.

Everyone is willing to accept this high level view. But what does this mean? Up to now, it is not quite clear yet how to make this vision work. What are experiences? Humans always use this word intuitively without defining its meaning. However, there are several possible definitions. But each definition will have a different impact on the derived activities. In the context of this paper experience is understood as *'collection of witnessing and insights gained by a human'* [6]. Strictly speaking, this implies that not experience itself (tuple of witnessing and insight) but experience knowledge (the insight) can be transferred. For the sake of simplicity the term 'experience' is used instead of 'experience knowledge' throughout this paper.

The next question will be 'How can experience be captured, documented, stored?' Experience has to be externalized to go beyond individual and towards organizational learning. To do so, a set of interacting mechanisms for experience acquisition, experience packaging, experience evolution, and experience reuse is needed.

In this paper, an implementation of these activities originating from the project SoftQuali is presented. Its main idea is an aggregation of knowledge from concrete *descriptive* observations to generalized *prescriptive* descriptions. We start the experience transfer process by capturing real-life observations in *observation packages.* An observation packages contains the description of a real-life fact together with a description of the context the observation was made in. In a subsequent step, we can use these observation packages to build more mature *practice packages*. Unlike observation packages, practice packages provide constructive descriptions of processes (e.g. how to perform inspections) or products (e.g. software architecture for ATMs). These descriptions are build upon the concrete observations provided by the observation packages and (potential) external knowledge. By this, the descriptions are tailored towards one environments needs and restrictions. On request, practice packages are provided with or without experts help to all advice-searching people in one enterprise. Figure 1 depicts the entire process graphically.

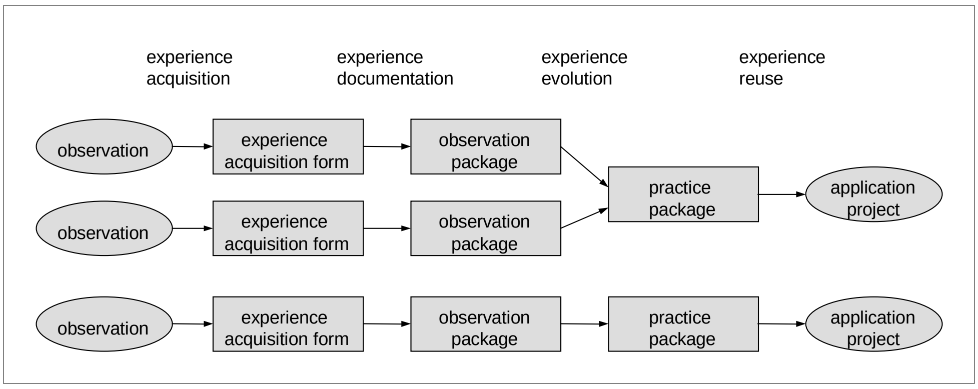


Figure 1 Process of experience acquisition, documentation, evolution, and reuse.

To document experience itself, we applied the Quality Pattern approach [4]. Quality Pattern support structuring and documenting of experience. Its main idea is a presentation of experience in pairs of problem and solution assuming the following experts' behavior: Every time an expert has to solve a new problem, she remembers similar, previously solved problems and adapts and combines those solutions to a new one.[1]

In the subsequent sections, the elements of this process are described in more detail. To make the process more comprehensible, two real-life examples are presented, too. Both examples are concerned with process improvement in the domain of software inspections. In the first one, documented experience to transfer this technology from one environment to another was used. In the second example, documented experience was used to improve inspections across several sites.

The main findings of the work presented here can be summarized as follows:

- *Quality Patterns and their subtypes are suitable for documenting various types of experience (observations, lessons learned, proposed best practice)*
- *The evolutionary quality pattern model (which is the maturation from observations to practice packages) supports the evolution of knowledge over time. It helps make the complex area of experience management both more understandable and more usable.*
- *The approach is applicable in real-life, as shown in the two application examples.*

Knowledge and experience are increasingly becoming the main assets of software developing companies. In order to keep or improve competitive advan-

[1] In this sense, this approach is quite similar to case-based techniques in artificial intelligence, see e.g. [5]

tage, establishment of successful experience transfer mechanisms is becoming crucial. The approach presented in this paper is the first step in this direction

The remainder of this paper is structured as follows. In Section 2, the quality improvement paradigm (QIP) and the results of the PERFECT project are briefly introduced as basis for the work presented in this paper. Additionally the project SoftQuali, its objectives, structure, and partners are described. In Section 3, the question of how to structure and document experience to make it reusable is emphasized. Sections 4 describes the task of experience acquisition. Section 5 gives a short glance at tool support for these activities. In Section 6, the application of the approach, in the domain of software inspections, is presented. Finally, we conclude in Section 7.

2 Context

2.1 QIP and PERFECT

Reuse of experience is a key element of the quality improvement paradigm (QIP) of Basili et al. [1]. In their work, Basili et al. describe a high-level process performed continuously. There are several steps in this process: First, an organization is characterized to determined their strengths and weaknesses. Based on these findings, improvement goals are defined and augmented by improvement procedures. During project work, these procedures are performed and measured. At the end of one cycle, the measured data is analyzed according to the defined goal and the findings are packaged for future usage. In their work, Basili et al. emphasize the importance of experience packaging. But they do not give any concrete guidelines for this task. Instead, they give some ideas on what experience packages should look like. For instance, they emphasize the importance of context descriptions, or introduce several types of experience packages, like *process packages* (experience packages providing experience on processes), *product packages* (e.g. software architectures), or *data packages* (providing quantitative materials, like fault distributions or cost estimation models).

The project PERFECT, founded by the European community, tried to make the idea of QIP more concrete [7]. Therefore, they were also concerned with the activities of experience documentation, evolution, and reuse. One result of the PERFECT project was a structure for experience packages consisting of the following parts:

- *a process model, describing a process for a particular task,*
- *process quality models, providing quantitative models on, e.g., effort, fault density, or size, and*
- *process experiences, providing lessons learned regarding the proposed process.*

The main deficit of this structure is that it is limited to process experience only and there exist no reports on its application. So it is unclear whether this approach really works.

2.2 The SoftQuali Project

The experience transfer approach presented in this paper is embedded in a larger structure within the SoftQuali project [8,15]. However, there are interdependencies between the various elements of this project. To make the presentation more understandable, first a short overview on SoftQuali is given.

The SoftQuali project (full title: systematic *soft*ware *quali*ty improvement by goal-oriented measurement and explicit reuse of experience knowledge) aims to apply the experimental paradigm (see, e.g., [14]) in the domain of software development and improvement. Figure 2 gives a graphical overview of this idea.

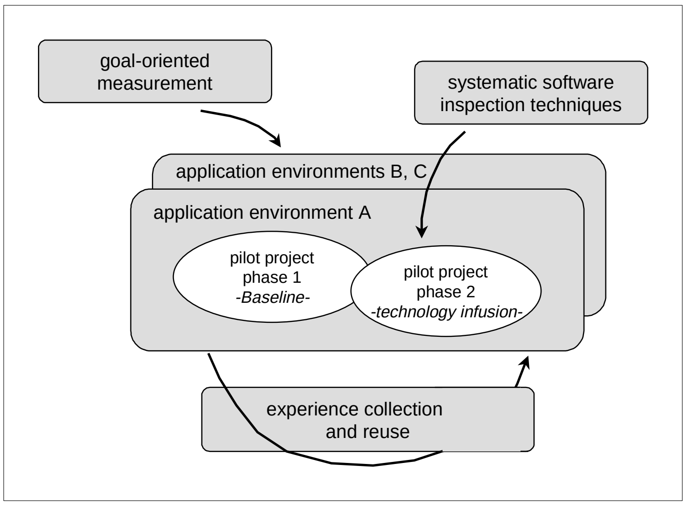

goal-oriented measurement
systematic software inspection techniques
application environments B, C
application environment A
pilot project phase 1 -Baseline-
pilot project phase 2 -technology infusion-
experience collection and reuse

Figure 2 Structure of the SoftQuali project.

In each of the application environments (depicted as A, B, C) the first activity was to build a baseline of the actual situation using goal-oriented measurement (GQM, [2]). On top of this baseline systematic software inspection techniques were tailored towards the needs of each application environment and introduced. Then, the new situation was measured as well. This whole process is augmented by experience collection and reuse to make the findings persistent and applicable both for new application environments within one partner and across the partners.

The partners in the SoftQuali project are

- *Allianz Life AG*
- *ALSTOM Energy Technology*
- *Daimler-Chrysler AG*
- *Fraunhofer Institute for Experimental Software Engineering (IESE)*
- *Siemens AG*

3 Experience Documentation and Evolution

3.1 **Types of Experience**

Documentation is essential to externalize knowledge and experience in order to enable people-independent transfer of experience. Especially in environments with a high turnover-rate (as in the software business, see, e.g., [12]) or in distributed development, documenting experience becomes a crucial activity.

Using the definition of experience given in the introduction, we can package experience in pairs of observations and conclusions. But this can be unsatisfyingly, as the subsequent example shows.

Example: To document the requirements for ATM switches, company XYZ uses the SA/RT according to Harel [19]. The project manager of this project observes, that this method is not successful in her project.

Here the question arises, whether the observations and conclusions should be described (e.g., 'SA/RT according to Harel is not suitable for analyzing ATM switches in company XZY because ...') or the proposals for improvement (e.g., 'I suggest to document requirements for ATM switches in company XYZ by ...'). Both alternatives are, in a way, problematic: In the first case, they document 'real' experiences, which may be of little use for future users ('Ok, I know what to avoid, but not, what to do'). In the second case, the proposed solution has not been *experienced.* We observed this dilemma quite often in our early work on documenting experiences. To deal with this problem, we build an evolutionary model of experience documentation including several types of experience packages:

- *Observation packages are used to package experience originating from real-life observations. They may, but need not, include conclusions on these observations. By this, observation packages are not necessarily useful for someone looking for a solution for his or her problem. But they are valuable input for subsequent activities, in which more mature experience packages are build.*
- *Practice packages provide applicable solutions for particular problems. These solutions must not necessarily have been experienced. They may originate from subjective conclusions or are derived from positive observations. The described solution is tailored for a particular environment, reflecting its situation, limitations, and special needs. The mission of a practice package is*

to provide the best[2] available solution for a problem in the given environment.

Another aspect which cannot be handled sufficiently using these experience packages concerns *external experience*. External experience is defined as knowledge that was gained in external environments and that is often generalized. Typical examples for external knowledge are the SA/RT technique and inspection guidebooks. These documents provide valuable information, but it is unclear what results the application of these descriptions will cause in a particular environment. To cope with this issue, theory packages .are introduced.

- Theory packages *provide external knowledge (i.e., experience gained in other environments). Typically, the information provided is very general and not adopted to the given environment.*

3.2 Evolution of documented experience

Using these types of experience packages, the evolution of documented experience can be illustrated quite easily (see Figure 3). Typically, it starts with external knowledge. Then this knowledge is tailored towards the special characteristics, constraints, and demands of the environment, building a first practice package. The application of this 'best practice description' will result in observations, which are packaged in observation packages. These observations may provide both positive and negative experience. If deficits are found in the previous solution, new insights can be used to improve practice, building a new, more mature practice package. This evolution of practice packages is going on continuously, using new observation packages as well as new theory packages.

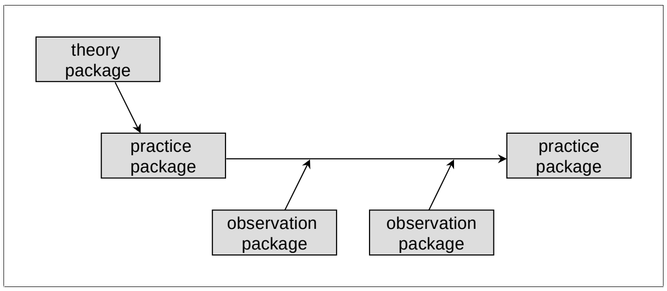


Figure 3 Evolution of documented experience

[2] It is important to see that the best available solution is time-dependent, i.e., the solution is not globally optimal, but optimal with respect to the current understanding. At a later point in time, another solution may be the best available solution.

3.3 Quality Pattern

Given the model of evolution of documented experience, the question of documentation has to be dealt with. However, plain documentation is not sufficient enough. Documented experience must be processed to be easily accessible, readable, and understandable. Aiming towards those requirements, the Quality Pattern approach [3,4] was developed. The main concepts of a Quality Pattern are problem-solution patterns, domain descriptions, explicit rationales, and the pyramid thinking principle.

The idea of problem-solution patterns was first published by C. Alexander [16] and has found wide adaptation in the area of design patterns. It is based on the observation that experts, who have to solve a new problem remember similar problems and adapt and adopt them to the new situation. They do not start from scratch every time. Quality Patterns use this idea by structuring information in a problem-oriented way. By this, the content of a Quality Pattern is aimed to fit a problem an experience-searching person will probably have (e.g., performing an inspection meeting, starting a new project, estimating project costs).

The description of the domain the experience was gained from is another crucial element in each Quality Pattern. The main reason for this is the understanding that no solution can be universally valid but is limited to a particular domain. The assumption is that applying a solution that worked for a *similar* problem in a *similar* domain will produce *similar* results.

By providing a rationale for the described problem-solution pattern, the reader can understand the *why* of the proposed solution and relate it to its own situation.

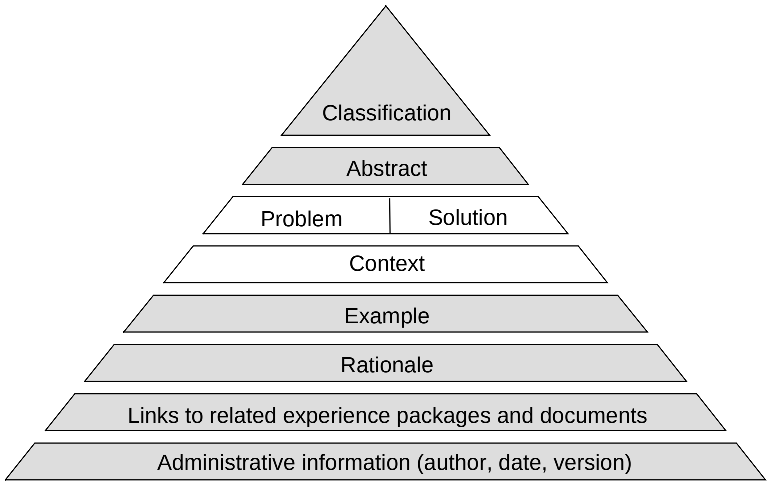

Classification
Abstract
Problem
Solution
Context
Example
Rationale
Links to related experience packages and documents
Administrative information (author, date, version)

Figure 4 Structure of a Quality Pattern.

The application of the pyramid thinking principle [17] produces documents which are structured in several layers. In the top layer, only the most important information is presented. The layers below cover more and more details without providing totally different information than the layers above. The advantage of this principle is that the main content of a document can be understood quite fast. There is no need to read a whole document to see that is does not provide any interesting information. Figure 4 gives a graphical overview of the different parts of a Quality Pattern.

The upper part contains summary information which help a reader to get a first impression of the Quality Pattern's content. Then follows the main part providing problem, solution, and context (domain description). The lower part contains more detailed information like an example, the rationales, and administrative information.

According to the evolutionary model mentioned above (Section 3.1), there are three subtypes of this generic quality pattern structure, called observation pattern, practice pattern, and theory pattern. They differ somehow in the various sections of a quality pattern. For instance, the section 'solution' in an observation pattern can be refined to 'observation' (what did really happen) and 'hypothesis' (what may be the reasons for it).

Figure 5 and 6 provide examples of a practice and an observation pattern, respectively. The next step using these pattern could be the creation of a new practice pattern describing the process of Figure 5 augmented with the strong recommendation to distribute the artifact under inspection via paper in the kick-off meeting.

Classification	Experience package type: Practice pack. Provided experience: Process Object: Reviews and inspections Problem: Kick-off meeting User: Inspections manager
Abstract	An inspection kick-off meeting consists of three steps: (1) distribution of review material, (2) explanation of inspection process, and (3) introduction of review material.
Problem	How do I perform an inspections kick-off meeting?
Solution	There are three steps in a kick-off meeting, with the third step being optional. (1) The moderator distributes the review material, i.e., the artifact under inspection (AUI), reference documents, checklists, and forms for defect detection. (2) The moderator explains the goals of the inspection process. (3) (Optional) The AUI author explains his product and technical details.
Context	Environment XYZ
Example	Here is an agenda of an inspection kick-off: 5 min, greeting 5 min, distribution of documents 10 min, explanation of goals and process 10 min, presentation of the AUI 10 min, discussion
Rationale	Mainly IEEE 1028; step (2) is non-optional, because inspections are performed rarely in this environment so there is the need to recapitulate the inspection process each time.
Links	Exp. Package: inspections form
Admin. info.	Author: W.C.Linton, December 18th, 1998

Figure 5 Practice pattern example.

Classification	Experience package type: Observation pack. Provided experience: Process Object: Reviews and inspections Problem: Kick-off meeting User: Inspections manager
Abstract	Consistent printouts of the artifacts under inspection (AUI) are an essential criteria for successful inspection meetings.
Problem	How do I perform an inspections kick-off meeting, even with time-pressure
Observation	Event: Inspection meeting Dec. 20th, 1998 Due to a rapidly performed kick-off meeting, the AUI (C-Code) was distributed after the kick-off meeting electronically. Later, in the inspection meeting, every inspector had a different listing (due to different line counting in the various mail systems and different printers). This caused a lot of confusion, because the different line numbers the inspectors reported had to be reassigned.
Hypothesis	- Time pressure - Inexperienced moderator and inspectors - Electronic distribution Documents must be distributed via paper in the kick-off meeting
Links	Exp. package: kick-off process (Fig. 5)
Admin. info.	Author: W.C.Linton, 21st December 1998

Figure 6 Oservation pattern example.

4 Experience Acquisition

We used experience packages according to the evolutionary quality patterns within various application projects (see also Section 6). Here, we made several observations:

- *People like experience documented in quality patterns. Typically, these experience packages are quite short but very concrete. They provide immediately applicable solutions.*
- *The incorporation of new experience, which leads to maintenance activities in the 'experience base', is well manageable due to the evolutionary model.*
- *It is hard to write good quality patterns. Writing a quality pattern is time consuming due to the preparation of information towards the quality pattern structure. Writing the redundant parts (e.g., the abstract) is boring, too.*
- *Quality patterns cannot be written spontaneously. Instead, a writer has to think about the particular needs a future reader may have.*

Despite – or especially because of – the advantages of quality patterns in reading, the inhibition threshold to write a quality pattern is high. To lower both the writing efforts and the inhibition threshold, we defined an 'experience acquisition form', implemented both on paper and using web technology [13]. The main idea of this form is that it can be used spontaneously. The sequence in which information is gathered differs from the sequence information is provided in a quality pattern. In these forms, the writer is asked for information she is able to provide quite easily, like a concrete observation or the description of her particular context. Redundant information (like the abstract) is not covered at all.

The experience acquisition form is used by project members. Every time they observe something remarkable, they can fill out such a form. In a later step, the information gathered is reprocessed by experts towards quality patterns as described above. By this, a kind of quality assurance process for quality patterns is established, as several people are concerned with writing quality patterns and reviewing other people's work.

Experience acquisition forms are mainly used for capturing observations. Capturing practice packages in this way is not very useful, since practice packages are seldom built by one person, but rather by groups of several people (like software engineering process groups, SEPGs, or quality circles).

Figure 7 presents an experience acquisition form. The Arial-typed text and the marked boxes show how this form can be filled in. The example text was used to build the observation pattern is shown in Figure 6.

Observation on ☒ process, ☐ product, ☐ data	
Date: August 20th, 1998	Observer: Calvin Klein
Project: XYZ	Role of observer: Insp.-moderator
Particularity of the environment: Inspections have been introduced right before	
Observation: The object under review (C-Code) was distributed electronically. Every inspector was asked to print it out by himself. In the inspection meeting there was a lot of confusion, because the line numbering in the various printouts differed.	
Hypothesis (assigned with the observation) ☒ Cause: High time pressure (no time for previous printout) ☒ Consequence/solution: Distribution of printed material is mandatory for every kick-off meeting	
Task (which somebody is interested in, in this experience): Planning and performing an inspection kick-off meeting	

Figure 7 Eperience acquisition form.

In the context of the transferred technology used (i.e., software inspections), we applied paper-based experience acquisition forms. Overall 24 observations within several weeks only were collected. The amount of time to fill in one form was only a few minutes.

It is important to see, that experience acquisition forms do not replace experience documentation using quality patterns, but help to build quality patterns. They help to lower the threshold for building observations patterns. In the later step 'experience reuse', quality patterns are most preferable (see also discussion at the beginning of this section).

5 Reuse of Experience and Tool Support

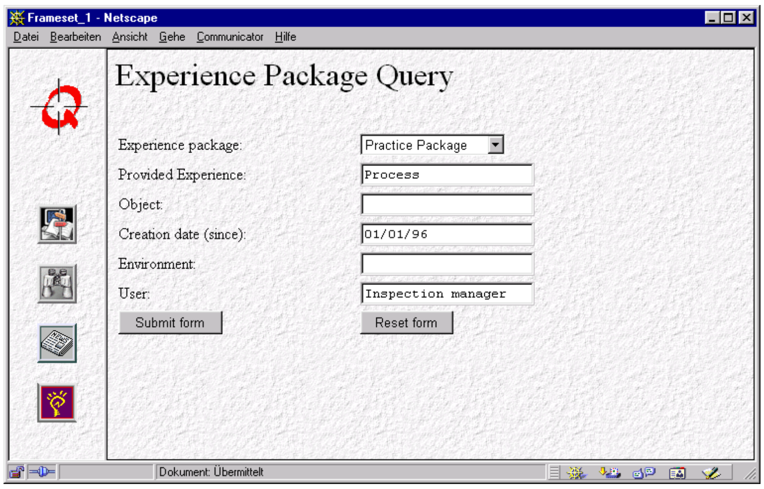
The main goal of the experience transfer process is, of course, the reuse of previously captured and packaged experiences. Users of experience (and experience packages) are, in general, people requiring advice. This may be

- *a project manager looking for experience on how to estimate a new project,*
- *a test responsible looking for information on using a particular test strategy,*
- *a developer who has to select a case tool for her new project and is looking for experience upon their utility in former projects, or*
- *a quality manager looking for quality assurance plans.*

Benefits of (re-)using (documented) experience are quite obvious. To reuse experience, it has to be delivered to the potential users. In our approach, this is done in two way. First, experience packages can be provided 'as is' by, e.g., electronic channels, like web or mail. This kind of experience transfer is mainly suitable for quite general information, like benefits of inspections in this company.

Second, experience packages can be provided in junction with human experts. In this case the expert brings the experience packages with him. They are present during the whole time the expert is working together with the project people. The expert himself uses them to do his job. Also, packages are improved or created during the interaction of experts and project people. This kind of interaction is preferable for more sophisticated material, like introduction of inspection processes. The main benefit of using experience packages during the interaction of experts and project people is, that after the departure of the expert the experience persists. And, it is not new to the project people (as e.g. a experts final report). Project people have seen how to use the information packages in the experience packages.

In addition, for electronic distribution we developed several prototypes [4]. They are all web-based in order to support different platforms in several sites. Figure 8 shows a screenshot of one prototype. It contains a search-mask for quality patterns. Attributes, which are left blank, are don't care attribute. If this form is submitted, beyond others, the pattern depicted in Figure 5 is received.



Experience Package Query

Experience package:	Practice Package
Provided Experience:	Process
Object:	
Creation date (since):	01/01/96
Environment:	
User:	Inspection manager

Figure 8 Screenshot of experience base prototype.

6 Application of the SoftQuali Experience Transfer Approach

In this section, two applications of experience transfer are presented. The first one is about transferring experience between several companies, the second one is about improving a technology by explicit documentation of experience.

6.1 Transferred Technology: Software Inspection

Both examples deal with inspections as transferred technology. Inspections have shown to be a very effective way of detecting defects early on in the development process. Thus, substantial effort can be saved, since defects detected later are known to be significantly more expensive.

However, there is not only one inspection technique, but several modifications of the idea of assessing a product by static examination. The implementations may differ, e.g., in the processes used, the roles involved, the artefacts inspected, and the reading technique used. In both examples the IEEE1028 definition of inspection is used, with one difference. In the first example, perspective-based reading (PBR), a new reading technique [9], based on the idea of TQM (Total Quality Management) [10] that each product has its own customers and that each customer views a product from his own point of view, was applied in the individual preparation phase and in the second example, some kind of checklist-based reading was used.

6.2 Experience Transfer between Fraunhofer IESE and Allianz

Situation. Development of high quality software satisfying cost, schedule, and resource requirements is an essential prerequisite for improved competitiveness of life assurance companies. Allianz Life, the market leader of life assurance companies in Germany, is part of the Allianz Group - one of the largest insurance groups with subsidiaries all over the world. The IT department of Allianz Life has more than 500 employees, with 350 of them being application developers. The Fraunhofer Institute for Experimental Software Engineering (FhG IESE) is acting as coach providing most up-to-date knowledge and experience on innovative reading techniques as the essential background technology.

Recent measurement programs have shown that currently quality assurance is focused on testing. At Allianz Life, about 30% of development effort are devoted to testing and about 50% of detected defects have their origin in early phases of software development. There, a technique is sought which improves productivity by finding defects when they are cheaper to correct, reduces the number of defects which originate in phases prior to the ones they are detected in, and reduces both test and overall project effort.

Use of documented experience. FhG IESE supported the introduction of PBR as a scenario-based reading technique at Allianz Life based on its wide range of experience in the field of software inspections. This step was accompanied by goal-oriented measurement to quantitatively demonstrate the proper usage of PBR as well as its cost/benefits ratio [18]. Additionally, experiences and measurement results are reused in the follow-up projects.

A post-mortem analysis of using PBR as defect detection technique at Allianz Life showed that not only the technique involved, or the organizational context, are the main driving factors for successfully introducing state-of-the-art technology. Another important factor were the experiences with that specific technique in different contexts. On the one side, this experience influenced the introduction process (what, when, and how to do) and gave confidence in the results to expect. On the other side, these experiences allowed inspectors to learn from others and use that newly gathered knowledge to improve their personal performance.

In order to support the introduction of PBR systematically, it is not sufficient to use that kind of experience described above on an ad-hoc basis, meaning the IESE-PBR expert reuses the knowledge stored in his memory. Instead, experience packages documenting different types of experiences, at different levels of abstractions, were used for planning and communication. Both observations packages and practice packages were created using the quality pattern structure. Consequently, each experience package contains a detailed description of the influencing factors which lead to that experience. A typical example for knowledge stored in an experience package is that ‘PBR-Scenarios for a specific organization can only be defined with the knowledge of a domain expert from that organization’.

Experiences on experience transfer. At Allianz Life, the use of experience packages was found to be beneficial for technology transfer in general and for the introduction of software inspections in particular. This was mainly due to the fact that typical problems could be avoided (e.g., involving too many people in an inspection meeting) or solved in advance (e.g., scenarios, the driving factor of PBR, were developed at the Allianz Life site). Furthermore, in comparison to technology transfer at other organizations without the help of experience packages, the effort was reduced significantly. This is also reflected by the experiences made in the HYPER project [18]. In Figure 9 for each inspection the total inspection effort is compared with effort savings in terms of ‘person hours’. It can be observed that for each inspection run the savings in effort are larger than the invested effort. We, Fhg IESE and Allianz, believe that this cost/benefit ratio was positively influenced by reusing documented, inspection experiences.

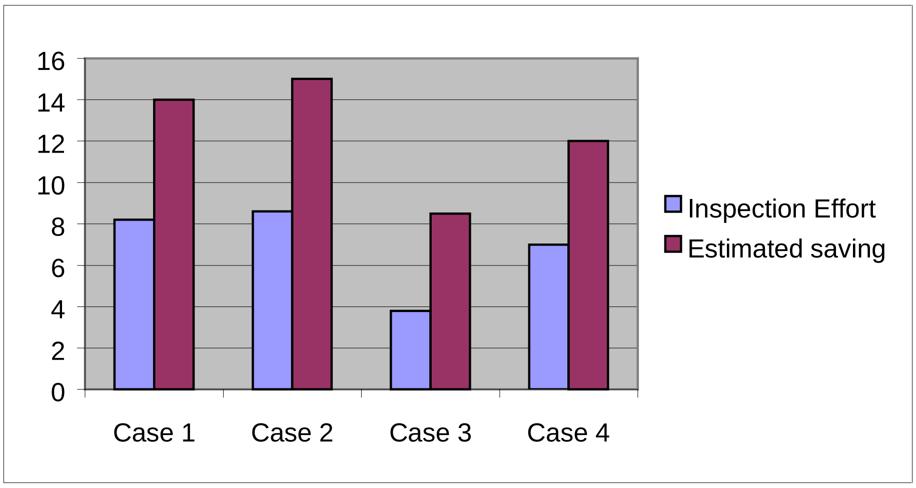


Figure 9 Primary Inspection Benefits [17]

6.3 Improvement of Inspections at DaimlerChrysler

Situation. Building mission-critical software at high quality within a tight schedule is increasingly becoming a core competency at Daimler-Chrysler. There are several divisions within Daimler-Chrysler (e.g., trucks or airplanes). And each division has its own software development unit, according to their particular needs. These units range from several dozens to several hundreds of software developers.

To improve the quality of their software development processes, the company's research division, introduced software inspections at several places. Unlike the Allianz-IESE approach, this happened mainly in the traditional way by providing teaching classes and coaching the inspection processes. This happened in several units quite simultaneously, with several researchers involved as coaches.

Use of documented experience. To improve both the introduction and the inspection process, and to avoid multiple pitfalls, experience documentation was incorporated in activities, starting in very early phases.

During their work in the application projects, the coaches documented their observations regarding inspections and the inspection introduction process. For this, they used experience acquisition forms as illustrated above. At regular meetings, the coaches discussed their observations and used them to build more mature practice packages which in turn they used in their daily work (see

Figure 10. In the following the content of these practice packages is illustrated through some examples:

- *Inspection process: In one unit, it is now mandatory to inform the inspectors about the goals of an inspection in each kick-off meeting, because there are quite long delays between single inspections)*
- *Forms: Inspections are now supplemented by role-specific inspections folders containing all material the various roles need (e.g., inspector: AUI, preparation form, inspection guidelines, checklist, experience acquisition form)*
- *Entry criteria: It is now mandatory in one unit to distribute the AUI on paper in the kick-off meeting (see also Figure 6 and 7).*

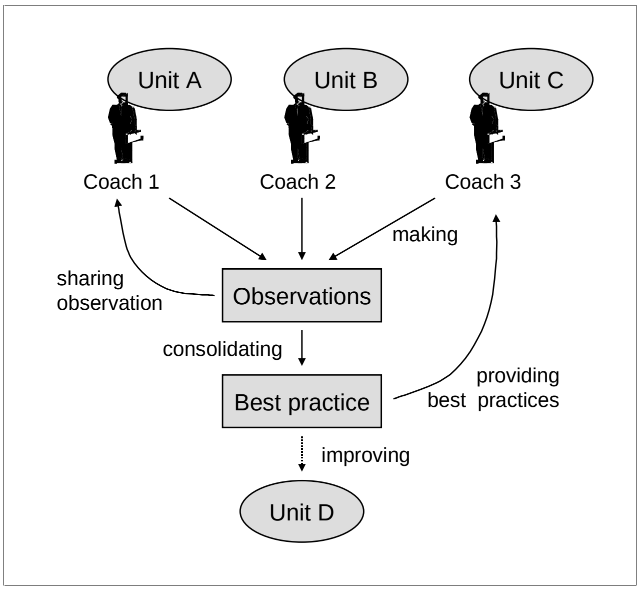


Figure 10 Using observations and experience to improve software inspections at several sites.

Experiences on experience transfer. The usage of observations packages (build by using experience acquisition forms) was found to be very useful. Without them, a lot of those typically small, but valuable insights in the inspection process would have been lost. This is especially important for mistakes everyone makes in the beginning when using inspections. If they would not have been packaged, some pitfalls could have not been avoided.

Another benefit is aggregating observations to practice packages. They provide not only applicable solutions but also rationales (in Section 'explanation' of the quality pattern structure). For smaller process changes, especially, the risk of reinventing the wheel twice (i.e., a change A, introduced first, is skied after a while by change B, which is replaced by change A, and so forth) was observed several times.

As in every technology transfer activity, inspection know-how is handed over gradually to the involved units. There, we see a benefit of the experience packages, too. Experiences described in such packages is:

- very concrete. *The content of the experience packages is tailored to the specific needs and restrictions of one domain.*
- built on experience in the user's environment. *By this, the practitioners in the units are willing to accept them more than external knowledge, as they were involved in building the experience packages.*
- quite short. *Due to high time-pressure no one is willing/able to read huge textbooks or conference proceedings.*
- maintainable. *Each unit can improve their processes using the experience evolution process described above (Section 3.1).*
- transferable. *They can both be used to make new employees familiar with the established processes and to exchange experience with other units in order to improve their overall understanding.*

The coaches will use the documented experiences both to train new researchers and to establish inspections in new units (as depicted by unit D in Figure 10). Starting on a more mature foundation of knowledge and experience, many troubles and pitfalls can hopefully be avoided.

7 Summary and Future Work

With the rapid rate of innovation in and around software technology, one might have expected to have seen significant improvements in the area of organizational learning and reuse of experiences. In practice, however, this has only started to begin due to the problems in capturing and reusing knowledge. The recent advent of experience factory technology [1] and its accompanying technology seems to be a promising approach to overcome those problems. In this paper, we have presented the experiences made with that technology in the SoftQuali project.

We described the basic technology of experience documentation and evolution. We introduced a crisp classification of experiences in three different types, providing a generic pattern (i.e., Quality Pattern) to document such types of experiences, and presented a practical approach for acquiring experiences. Finally, we presented the results of using this technology in two real-world examples, both related to introducing inspection technology. In general, the reuse of documented knowledge/experience to plan and control the software technology transfer process was found to be useful in both projects.

Our next activities cover several directions. First, we see the need for gaining more practical experience in using our approach in several domains. Then, we plan to build more sophisticated tools to support humans in their experience maintaining activities. Last, but not least, we need to evaluate the quality pattern structure more thoroughly. Especially our assumption that quality patterns are more comprehensible and written towards a readers needs, has to be validated empirically. As a first step here, we are currently performing a long-term study in a laboratory environment at the University of Ulm. There, we have asked students to write both quality patterns and reports on particular topics (namely inspections and modeling reactive systems). At a later point in time, we will give these documents to other students, asking them to perform these particular tasks and observe the time and depth of understanding. Subsequently, we plan to use these findings to start such an investigation in an industrial setting.

Acknowledgements

The authors would like to acknowledge the contributions of all SoftQuali partners. Thanks are also extended to Sonnhild Namingha for her efforts in editing this paper.

This work was founded by the German Federal Ministry of Education, Science, Research, and Technology under contract number 01 IS 518. It does not necessarily reflect the position or the policy of the government and no official endorsement should be inferred. The responsibility for the content of this paper remains with the authors.

References

[1] V.R. Basili, G. Caldiera, and H.D. Rombach. *Experience Factory*. In J.J. Marciniak (ed.): *Encyclopedia of Software Engineering*. John Wiley & Sons, New York, 1994, pp. 469-476.

[2] V.R. Basili, G. Caldiera, and H.D. Rombach. *Goal question metric paradigm*. In J.J. Marciniak (ed.): *Encyclopedia of Software Engineering*. John Wiley & Sons, New York, 1994, pp. 528-532.

[3] D. Landes, K. Schneider, and F. Houdek. *Organizational learning and experience documentation in industrial software projects*. In Proceedings of the 1st Interdisciplinary Workshop on Building, Maintaining, and Using Organizational Memories (OM-98), Brighton, UK, August 1998, pp. 47-63.

[4] F. Houdek and H. Kempter. Quality Pattern – An approach to packaging software engineering experience. In M. Harandi (ed.): Proceedings of the 1997 Symposium on Software Reusability (SSR'97), Software Engineering Notes, ACM, Mai 1997, pp. 81-88.

[5] M.Y. Jona and J.L. Kolodner. *Case-based reasoning*. In S.C. Shapiro (ed.): *Encyclopedia of Artificial Intelligence*. John Wiley & Sons, 1987, pp. 1265-1279.

[6] Lexikon-Institut Bertelsmann, *Dictionary*, Bertelsmann, Gütersloh, 1979 (in German).

[7] *The PEF model*. A booklet from the PERFECT EPSRIT project 9090, 1996.

[8] The SoftQuali Consortium. *SoftQuali Project Documentation*. Available at http://www.iese.fhg.de/ Proj/SOFTQUALI/index.htm.

[9] V.R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S, Sorumgard and M.V. Zelkowitz. *The Empirical Investigation of Perspective-Based Reading*. Journal of Empirical Software Engineering, 1997.

[10] Richard E. Zultner. *TQM for Technical Teams*. Communications of the ACM, 36(10), October 1993, pp79-91.

[11] Adolf-Peter Bröhl and Wolfgang Dröschel. The *V-Model*. Olden-

bourg-Verlag, 1995 (in German).

[12] T. DeMarco and T. Lister. *Peopleware*. Prentice Hall, 1995.

[13] W. Bierer. *A Process for Building Quality Patterns*. Masters thesis, University of Stuttgart, May, 1997 (in German).

[14] H.D. Rombach, V.R. Basili, and R.W. Selby. *Experimental Software Engineering Issues: critical assessment and future directions*. Lecture Notes in Computer Science 706, Springer, Berlin, 1993.

[15] H. Kempter and F. Leippert. *Systematic Software Quality Improvement by Goal-Oriented Measurement and Explicit Reuse of Experience knowledge*. BMBF-Statusseminar 1996, pp. 281-297, DLR (German Center for Airospace, in German).

[16] C. Alexander. *The Timeless Way of Building*. Oxford University Press, Oxford, 1979.

[17] B. Minto. *The Pyramid Principle - Logic in Writing and Thinking*. Minto International, London, 3rd edition, 1987.

[18] L. Briand, B. Freimut, O. Laitenberger, G. Ruhe, and B. Klein. *Quality Assurance Technologies for the EURO Conversion – Industrial Experience at Allianz Life Assurance*. Proceedings of the Software Quality Week Europe, Brussels, 1998.

[19] D. Harel. Statecharts: A visual formalism for complex systems. Journal of Science of Computer Programming, vol. 8, 1987, pp. 231-274.

Document Information

Title:	Transferring Experience: A Practical Approach and its Application on Software Inspections
Date:	February, 1999
Report:	IESE-008.99/E
Status:	Final
Distribution:	Public