Fraunhofer Einrichtung
Experimentelles
Software Engineering

A Knowledge Management Lifecycle for Experience Packages on Software Engineering Technologies

Technical Report

Authors:
Andreas Birk
Felix Kröschel

IESE-Report No. 007.99/E
Version 1.0
February 1999

A publication by Fraunhofer IESE

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

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

Abstract

Software engineering can benefit very much from customised knowledge management solutions. These should rely on reusable experience that is modelled explicitly and stored in central repositories. Few approaches exist yet that provide such knowledge management support to software engineering. Those that support it cover usually only part of the knowledge management lifecycle of the reusable artefacts.

This paper suggests a knowledge management lifecycle for experience about software engineering technologies and their application contexts. It primarily aims at supporting the planning of software projects and improvement programmes. The lifecycle model is substantiated by a tool implementation and evidence from an industrial trial application.

Table of Contents

1 Introduction

Software engineering plays a key role in today's public life. Nearly every product of the manufacturing or service industries depends to a wide extent on software. The source of software engineering's impressive success is a tremendously fast progress in developing and deploying new technology (cf. [You93]). However, despite these achievements, stories about computing problems and failures are spread over the news quite frequently. These troubles seem not to fit into the picture of a successful new branch of engineering. A number of recent investigations have shown that the main cause of software engineering's failures is the same as for its successes: New technology–and, in the case of project failures, the inability to manage it successfully (cf. [Sta95], [KPM95], and [Gla98]).

The inherent risk of technology failure in software projects calls for a better management of our knowledge about technologies and their application contexts. Since the beginning of software engineering much effort has been put into the development of new technologies. Less attention has been paid to their application and empirical evaluation. We argue that this lack of information about when a certain technology can be applied most appropriately and when it should not be applied is the main reason for many technologies-caused project failures.

A knowledge management approach to the application of software engineering technologies has two basic requirements: (1) Precise and operational definition of software engineering technologies, and (2) the systematic investigation and documentation of the application contexts of technologies. While the first issue has already been subject to many research efforts, mainly in the area of process modelling (cf. [RV95], [CKO92]), the application contexts of technologies have

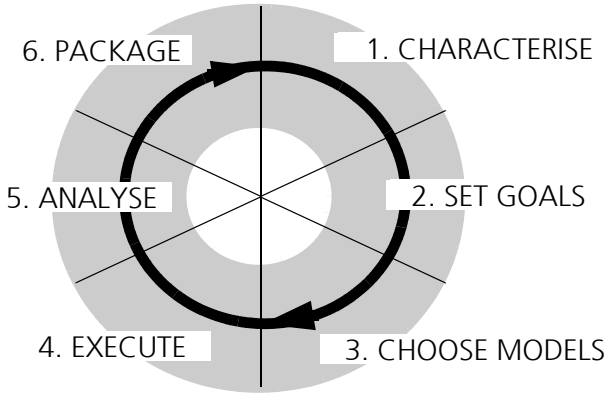

Figure 1: The Quality Improvement Paradigm.

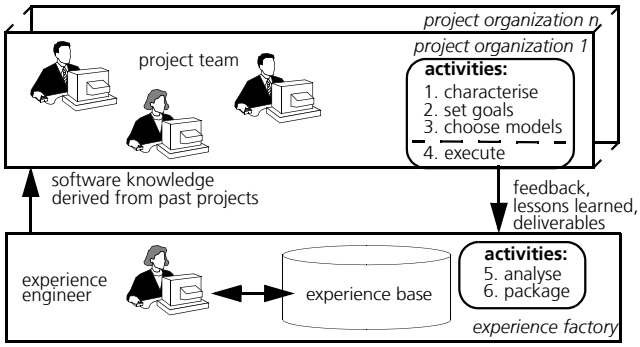

Figure 2: The Experience Factory.

hardly been addressed, yet. For this reason, we have been focusing our research around the following questions: How can technology application contexts can be modelled?, and how can concrete context models be gained, used for supporting project planning, and evolved based on experience from technology application? This paper presents results from this work. It introduces so-called *technology experience packages (TEPs)* for modelling. Further, a lifecycle model for developing, using, and maintaining TEPs is presented. For each lifecycle task possible support is illustrated using a prototypical tool implementation.

Our approach builds on the Quality Improvement Paradigm (QIP) / Experience Factory (EF) approach [BCR94]. The QIP is a six-step cycle for continuous improvement in software engineering (see Figure 1). It serves also very well as a knowledge management paradigm for the software domain (cf. [TA97]). The EF is the associated infrastructure for organisational learning (see Figure 2). It distinguishes the project organisation, whose main concern is software development, from the Experience Factory, whose main concern is to learn about software development and to support the project organisation with useful experience. Core component of the EF is the *Experience Base (EB)*, a repository of relevant software engineering experience. The EB contains a collection of *experience packages* that consist of a reusable artefact (e.g., a process model, an effort prediction model, or a code module) and information about when–i.e., in which situations–the reusable artefact can be applied.

Throughout this paper, focus is put on software engineering technologies. Therefore, we refer to a specialised variant of experience bases: Technology experience bases (TEBs), that contain technology experience packages (TEPs). As software engineering technology we refer to every technique, method, or tool used for software engineering [BCR94]. Special attention is paid to process technologies, such as inspection methods and methods for performing measurement programmes.

Section 2 addresses the interface between knowledge management and software engineering. It provides a survey of selected knowledge management solu-

tions from software engineering and other fields. The concept of *technology experience bases* is introduced in Section 3. Sections 4 to 7 present a lifecycle for managing knowledge about software engineering technologies and their application domains. A project on the development of a TEB and its experiences are reported in Section 8. Section 9 presents the main conclusions from our work.

2 Knowledge Management and Software Engineering

Knowledge management can provide beneficial support to software engineering (cf. [BSA99], [Eri92]). This section explores the interface between these two areas. First, requirements on knowledge management of software engineering technologies are identified (Section 2.1). These are then used in Section 2.2 to survey and characterise relevant knowledge management tools. Finally, conclusions are derived from this survey that pin-point further needs for methodology and tool support.

Knowledge management is a term that is still widely discussed and that has no comprehensive and widely accepted definition yet. A concise definition that is widely appropriate for software engineering follows a definition proposed by O'Leary [OL98']:

> *Knowledge management is the formal management of knowledge for facilitating creation, access, reuse of knowledge, and learning from its application, typically using advanced technology.*

This definition reflects the need of a learning organization for identifying knowledge, for storing it appropriately, for making it accessible, and for easing its reuse. Experience gained from reusing the stored knowledge items should be deployed to further evolve and extend the stored body of knowledge.

2.1 Requirements on Knowledge Management of Software Engineering Technologies

The need for knowledge management support in software engineering–in particular for the management of knowledge about software engineering technologies and their application contexts–can be specified in the form of requirements. In the following, a set of such requirements is provided. Some of these have been derived from the literature ([Hen97b], [BR91]). Others stem from an analysis of knowledge management needs concerning software engineering technologies (cf. [BSA99]).

Support the entire lifecycle of reusable artefacts

A knowledge management system should support the entire lifecycle of the managed artefacts (or experiences or knowledge items), involving their creation, their retrieval and reuse, as well as the feedback of experience from using them.

Similarity-based retrieval

Retrieval of reusable artefacts should be based on appropriate similarity measures that do not require exact matches between a retrieval query (i.e., the specification of reuse requirements) and the retrieved reusable artefacts. This is necessary for knowledge reuse in software engineering, because the characteristics of software projects can differ quite much. So it is very likely that relevant reusable artefacts are characterised in a form that is not exactly matching a reuse situation.

Retrieval based on incomplete information

Retrieval of reusable artefacts should not require that for all attributes using which the artefacts are indexed information is provided in the retrieval query. This is necessary, because knowledge retrieval is usually performed at the beginning of a software project when some characteristics of the forthcoming project are not yet known. It should be possible to omit these characteristics when defining a retrieval query.

Possibility to evolve the experience base

The TEB should provide mechanisms for adding new knowledge artefacts, as well as for adding or removing attributes of existing objects. It should also be possible to generalise or specialise existing objects. This is necessary in order to support continuous learning and improvement in software engineering.

Support different types of knowledge

The experience base should support a variety of different knowledge types. They can have different representation structures, different access and usage processes, as well as different levels of maturity and reliability.

Precise representation of reusable artefacts

Reusable artefacts should be represented and stored–as far as possible–using a precise and sufficiently formal knowledge representation. This is needed for clarifying the semantics of the artefacts as well as for allowing for automated support of knowledge usage and maintenance.

Intuitive characterisation of reusable artefacts

Reusable artefacts–and in particular their application context–should be defined using terms that are intuitive to the knowledge users. This is important to assure that the experience base is accepted from its users and to allow for efficient usage processes.

Table 1: Comparison of selected knowledge management tools

	Support the entire lifecycle of reusable artefacts	Similarity-based retrieval	Retrieval based on incomplete information	Possibility to evolve the experience base	Support different types of knowledge	Precise representation of reusable artefacts	Intuitive characterisation of reusable artefacts	Links between reusable artefacts
Bore	●	●	●	◆	◆	●	◆	●
Msmt Planning	◆	●	●	◆	◆	●	◆	●
KONTEXT	●	●	●	◆	◆	●	●	●
WebME	◆		◆	◆	◆	●	◆	●
IDS		◆	◆	◆	●	●	◆	
Ontobroker		◆	◆	◆	◆	●	◆	◆
Kactus			●	◆	◆	●	◆	◆
Dedal		◆	●	◆	◆	●	◆	●
QuestMap	❍			◆	●	●	◆	●
eule2	❍		◆	◆	●			●
LotusNotes/ Domino	❍	❍	❍	◆	●	❍	❍	❍
LiveLink	❍	◆	◆	◆	●			●
AnswerGarden	❍			◆	◆			●

● = comprehensive support ◆ = support exists in part ❍ = support is not explicit, but the tool offers some means to develop it

Links between reusable artefacts

Explicit links should be established between related artefacts in the experience base. This is needed for being able to exploit relationships during maintenance and retrieval. For instance, technologies should be linked with the processes in which they can be applied.

There are also additional requirements that are important for the technical realisation of an experience base. Examples are access via the internet or intranet, view support, and information security. These are not addressed further, because the focus here is on the methodological concepts of knowledge management in software engineering.

2.2 A Survey of Knowledge Management Tools

Many knowledge management approaches rely mainly on the organisation of knowledge exchange between human agents (cf. [Wii95] [OL98'] [Sen90] [SKR+94]). However, our objectives for knowledge management in software engineering are to make knowledge assets explicit (e.g., in the form of technology experience packages), to store them in repositories that are widely accessi-

ble (i.e., technology experience bases), and to provide computer support for the management of this knowledge.

There is quite a number of tools available–either as research prototypes or as commercial tools–that address knowledge management. Some of these are specialised to knowledge management in software engineering. Abecker et al. [ADK98] distinguish between process-oriented and product-oriented tools. Process-oriented tools aim at supporting or facilitating knowledge exchange between human agents during their work processes. Examples are groupware systems and intranet solutions. These tools are usually not particularly effective for supporting the management of knowledge about software engineering technologies and their application contexts. Product-oriented tools focus on the knowledge assets to be reused. These tools offer appropriate concepts for the knowledge management of software engineering technologies and application contexts.

Table 1 shows a collection of selected knowledge management tools and describes them with regard to the requirements stated in the previous section. The following tools have been included into the survey: *Bore* ([Hen97a] [Hen97b]) manages knowledge that supports the execution of corporate software development processes. It contains guidelines, lessons learnt, and frequently asked questions, following a case-based approach to knowledge representation and management. Gresse von Wangenheim et al. [CGvW98] are developing a tool for capturing experiences about GQM measurement planning (in the following denoted as *Msmt Planning*). *KONTEXT* [Krö98] is a tool for managing the knowledge about software engineering technologies and their application contexts. It is described in more detail below, in Sections 4 to 7. *WebME*, a web-based tool for providing measurement data to project management has been developed at NASA's the Software Engineering Laboratory ([RZ93] [TZ98]). These four tools are all knowledge management solutions that are designed for specific software engineering tasks. The following tools are designed primarily for other domains, or they provide generic knowledge management infrastructures.

The expert system *IDS* [IBM] has been developed for the aircraft industry by making use of data-mining techniques for diagnosing purposes. *Ontobroker* [FDES98] is an intelligent search tool (e.g. for searching the internet) based on ontologies. *KACTUS* [SWI97] provides an interactive environment for browsing, editing and managing (libraries of) ontologies. A representation of a device model is used by the tool *Dedal* [ded95] for indexing and retrieving multimedia information about a designed device. The hypermedia groupware system *QuestMap* (cf. [Shu97], [Sys98], and [CY93]) integrates different media and links the captured knowledge into a structure supporting the discussion process. *Eule2* [Rei97] is a knowledge-based system for supporting office work in the life insurance domain. It integrates knowledge bases of different knowledge-based systems. The collaboration process in an organisation is supported by the widely

used groupware tool *Lotus Notes/Domino* [Lot]. For instance, it provides services for electronic mails, storing various types of documents and information, and it allows for annotating documents. The tool *LiveLink* [Tex] incorporates elements of collaborative work support and web agents on a document-centred knowledge base. By using *AnswerGarden* [MT90] two different types of knowledge are combined: recorded knowledge can be retrieved and individuals with knowledge of some kind are made known to the rest of the organization. Information access is organised through posing diagnosis questions.

These tools and their underlying methodologies represent quite different philosophies of knowledge management. It was a purpose of the survey to provide a broad overview and to place the four SE-specific tools into the context of other kinds of knowledge management tools. Please note that the classification schemes (i.e., the symbols in the table cells of Table 1) of different evaluation criteria (i.e., the table columns) can have slightly different semantics. The survey is not meant to rank the tools. Its objective is to illustrate that there exists a variety of technical solutions to common knowledge management requirements.

2.3 Observations from the Tools Survey

The knowledge management tools characterised in Table 1 can be evaluated using the requirements listed in Section 2.1. This evaluation provides an overview of the state of the art in knowledge management for software engineering applications. This section identifies the observations from the survey and briefly outlines needs for further tool support.

Support the entire lifecycle of reusable artefacts

The four tools that are specific to software engineering provide specialised support for more than one lifecycle phase. *Bore* and *KONTEXT* have a usage model that comprises all lifecycle phases from insertion of new knowledge via knowledge use to knowledge evolution. Other tools do either address only the core phase of knowledge use (then nothing is indicated in the table), or they support some kind of knowledge evolution without modifying specific artefacts. The latter is especially true for process-oriented knowledge management tools such as workgroup support systems.

Similarity-based retrieval

Similarity-based retrieval involving some kind of similarity function is realised in *Bore*, *Measurement Planning*, and *KONTEXT*. Other tools realise similarity-based retrieval using some other concepts such as heuristics. *Lotus Notes/Domino* offers the basic infrastructur for implementing similarity-based retrieval mechanisms.

Retrieval based on incomplete information

Again, *Bore*, *Measurement Planning*, and *KONTEXT* use explicit characterisation schemes for specifying queries and allow queries in which some characteristics are omitted (i.e., retrieval with incomplete information). This is also true for *Kactus* and *Dedal*. Other tools don't use characterisation schemes for knowledge representation and retrieval, but they allow for some kind of open retrieval.

Possibility to evolve the experience base

Dynamic extension and modification of the experience base is a standard feature that is provided by all surveyed tools in that new knowledge items can be added, removed, or modified. Specific support for automated generalisation or specialisation is not provided at all.

Support different types of knowledge

Every tool represents multiple types or dimensions of knowledge. But only some tools do also allow for extending the set of pre-defined knowledge types or are fully open with regard to the managed knowledge types.

Precise representation of reusable artefacts

Most tools apply some formal or at least well-defined structured knowledge or data representation. But some, mainly the hypertext-based ones, do not have a particularly well-organised, finer-grained representation scheme.

Intuitive characterisation of reusable artefacts

KONTEXT is different from the other tools in that it offers a specific representation concept for characterising the reusable artefacts in an intuitive manner. This is done using a redundant characterisation structure that has one view which models the intuitive terms used by decision makers, while the other view provides a precise and unambiguous definition of context characteristics. Most other tools allow for using intuitive identifiers of knowledge items, but only as far as object identity can be assured.

Links between reusable artefacts

Most tools have some kinds of links between reusable artefacts thus that relations between associated kinds of knowledge can be explored by the user. These clusters of linked artefacts thus establish some new kind of "higher-order" object and allow the user to explore it interactively. Ontology-based systems (i.e., Ontobroker and Kactus) do of course also implement links between objects. But these objects are mostly of similar kind, and the linked entities do

not establish a really complex new knowledge structure (other than the ontology itself).

As overall conclusion from the survey it can be noted that there exists a wide variety of knowledge management solutions. Some have been designed specifically for use in software engineering, and these offer quite a wide spectrum of features. For several of the requirements, *KONTEXT* offers the most advanced support among these tools. *Lotus Notes/Domino* is potentially satisfying all requirements. But it would require some considerable implementation effort to actually establish these functions.

Future developments would be needed most in the areas of comprehensive life-cycle support (i.e., offering specific support for inserting new knowledge and evolving the already represented knowledge), support for automated generalisation and specialisation of artefacts[1], improved integration of multiple different knowledge types, and the definition of intuitive views on the stored knowledge, which can be tailored to the needs of certain user groups.

1 Basic technologies for this are provided for instance from the field of machine learning.

3 Technology Experience Bases

The technology experience base (TEB) is the basic repository of knowledge about software engineering technologies and their application contexts onto which our approach is based. It has been introduced briefly in Section 1. In the following, the structure of TEBs is explained in some more detail.

A TEB has basically two kinds of contents: (1) A collection of technology experience packages (TEPs) and (2) a collection of background definitions and taxonomies (see Figure 3). A TEP contains the definition of the respective technology (usually in the form of a process model) and specifies for which process (e.g., software design or measurement) it can be applied, which product quality (e.g., reliability or maintainability) of which product type (e.g., embedded control systems or medium-sized information systems) can be yielded using the technology, and in which context situation this application and quality impact of the technology can be expected (cf. [Bir97]). The context situation is defined through a set of *context characteristics*. A context characteristic is an attribute/value pair that defines a characteristic of a software project such as team size or

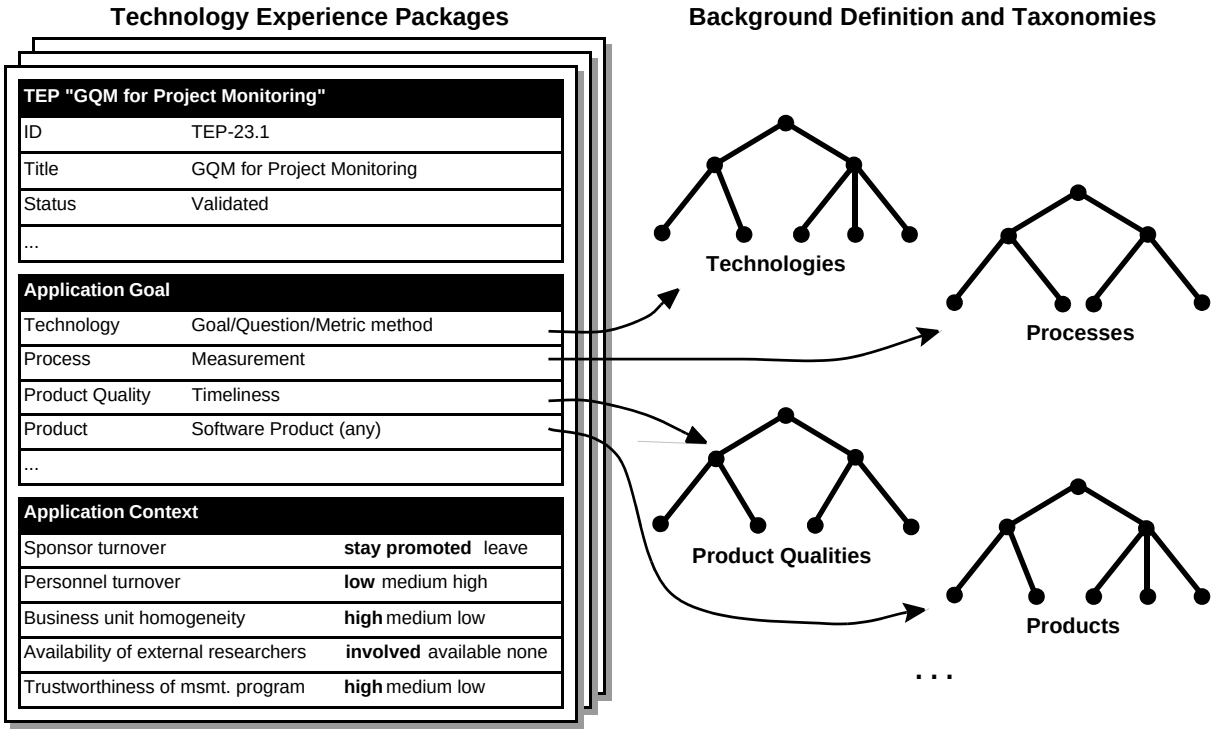

Figure 3: Technology experience packages and their relations to background definitions and taxonomies.

degree of management commitment. The attribute and its definition is referred to as *context factor*. A collection of associated context factors forms a *context model*.

Background definitions and taxonomies are needed for achieving uniform definitions of the contents of TEPs, for avoiding redundant storage of these TEP contents, and to provide an index structure for experience retrieval. Hence, each concept contained in a TEP has an associated background definition and is part of a taxonomy. A TEB contains–among others–taxonomies of technologies, processes, products, and product qualities.

4 A Knowledge Management Lifecycle for Technology Experience Packages

The tool survey in Section 2 has shown that most knowledge management approaches are lacking yet a comprehensive lifecycle support for the managed knowledge artefacts. In this and the following sections we use the case of technology experience packages to illustrate how such a comprehensive lifecycle support can look like. The approach is substantiated by examples from a specialised knowledge management tool.

The knowledge management lifecycle for TEPs involves three main phases (Figure 4):

- Gaining TEPs using *knowledge acquisition* and other techniques.
- Using TEPs to support software engineering, e.g., for *technology selection* during the planning of software projects or improvement programmes.
- Updating and evolving TEPs based on the *empirical evaluation* of software projects that have applied them.

In addition, we briefly address the topic of building and installing a TEB (Section 5).

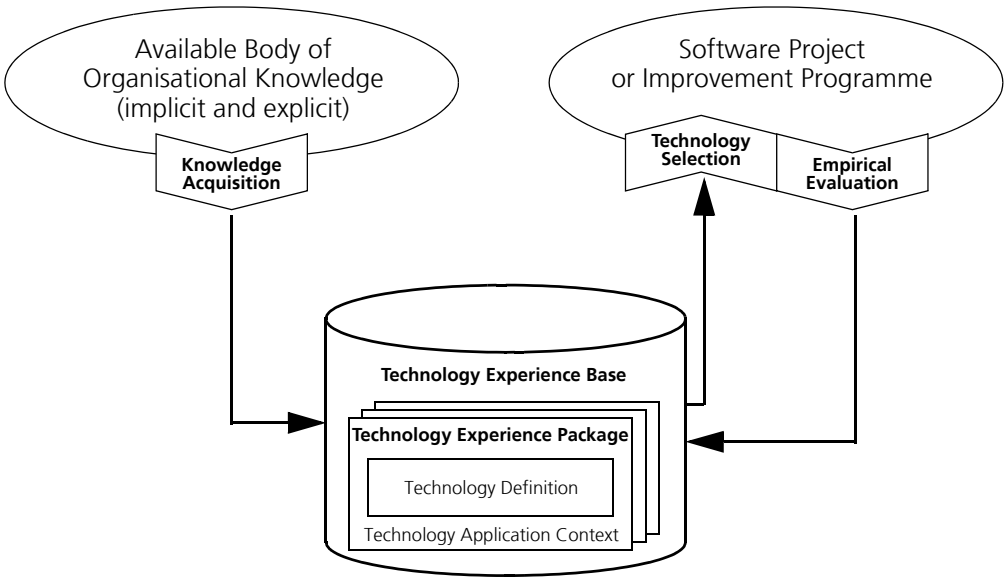

Figure 4: The lifecycle model of technology experience packages.

For gaining TEPs a particular focus is put on knowledge acquisition techniques, because we claim that much knowledge about the application context of software engineering technologies exists already implicit in the minds of experienced software professionals, or it is distributed over a large number of different media throughout a software organisation (cf. [BSA99]). Knowledge acquisition is considered to be the most beneficial source of knowledge when setting up a new TEB. The usage of TEPs and their empirical investigation are closely related to software projects or improvement programmes for which appropriate technologies are needed, and in which they are used.

A prototypical tool has been implemented for supporting the knowledge management lifecycle for TEPs. The tool is called *KONTEXT* (*KnOwledge maNagement based on the application conTEXt of software engineering Technologies*). It offers functions for the knowledge modelling tasks involved in knowledge acquisition and empirical evaluation of TEPs, as well as for the entire technology selection process. In the following sections, these functions of *KONTEXT* are explained in more detail.

5 Knowledge Acquisition of Technology Experience Packages

Most software organisations and experienced software professionals know much about the application contexts of software engineering technologies. However, this knowledge is often implicit or distributed over a large number of sources and hardly accessible. For this reason, knowledge acquisition is the preferred approach to quickly gain a large number of TEPs and to populate an initial TEB.

Alternative or additional information sources for developing TEPs are the software engineering literature, software measurement programmes, data mining and information research (using past project documentation, existing measurement data, or other organisational files), as well as empirical investigations such as surveys or case studies. However, all these techniques can also benefit from prior knowledge acquisition efforts that provide grounded hypotheses about the technologies and their application context.

The initial construction of TEBs, which should afterwards be extended and updated continuously, can in principle be done according to the following basic steps:

1 Identify and define the set of relevant technologies.

2 Determine the required target set of TEPs by specifying their processes, product types, and product qualities.

3 Conduct a pre-study of past projects in which the technologies have been used and collect relevant information from literature.

4 Develop background definitions and taxonomies of all relevant concepts that are to be represented in the TEB (e.g., precise operational technology definitions, process taxonomies, product quality definitions and taxonomies).

5 Conduct knowledge acquisition in order to gain context characteristics of the TEPs.

6 Model the TEPs based on the knowledge gained from knowledge acquisition.

7 Verify the modelled TEPs.

8 Validate the TEPs.

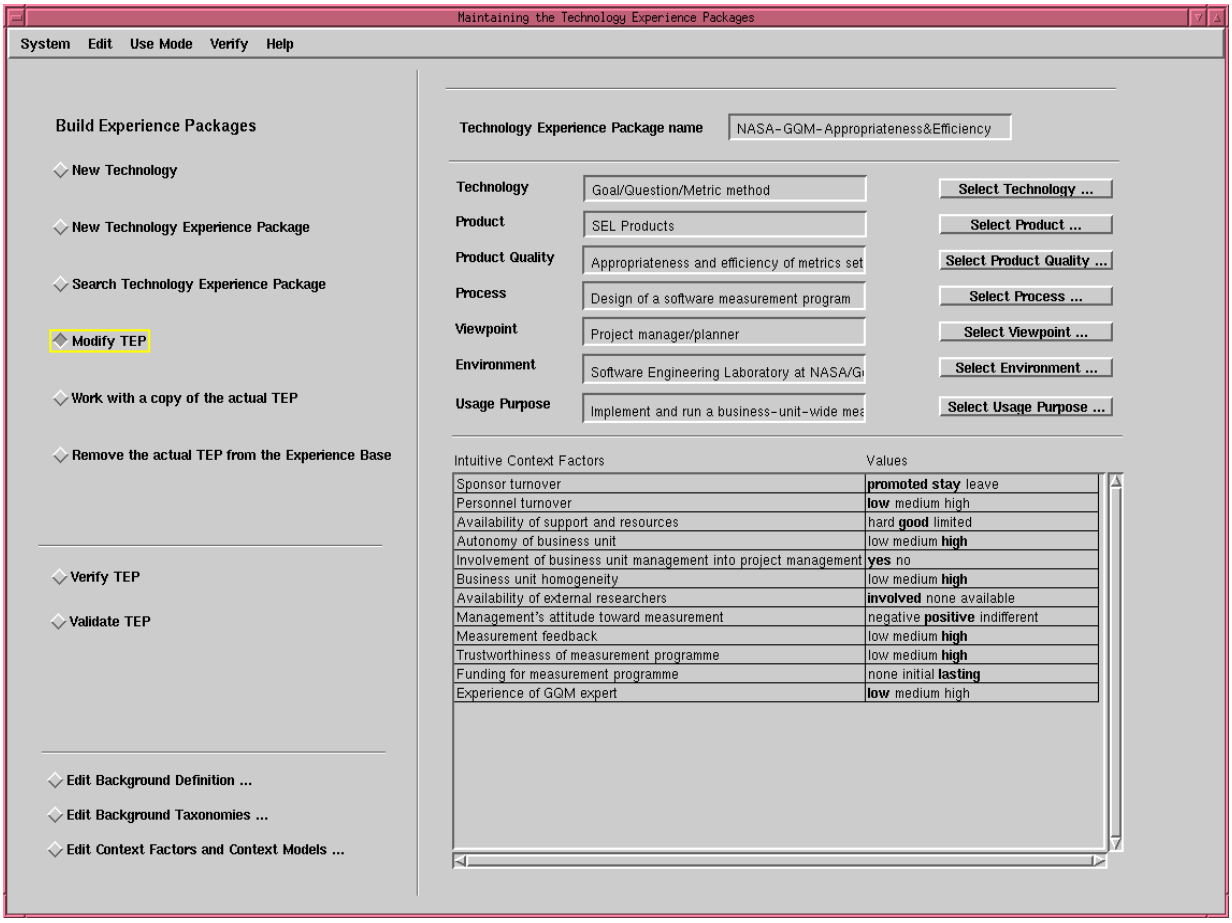

Figure 5: Modifying a technology experience package.

Figure 5 depicts the user interface of the TEP modelling component of *KONTEXT*. The leftmost part of the window guides the user through the process of modelling a collection of TEPs. It involves the insertion of a new technology, initialisation of a new TEP, modification of a selected or new TEP, as well as verification and validation of TEPs. The rightmost part of the window offers the functionality to perform the selected process step, in this case the modification of a selected TEP. The entire TEP is displayed and each component of it can be modified or extended.

Verification and validation of TEBs are very important. *KONTEXT* supports the automatic verification of TEPs such as completeness checks and some basic consistency checks. The extensive user guidance of *KONTEXT* avoids widely that wrong kinds of information could be inserted. Validation is currently supported through offering a well-structured report view on TEPs for inspection. Plans for future validation features involve several kinds of simulation-based and multi-expert checking.

6 Technology Selection Support Using Technology Experience Packages

A TEB can be beneficial for a multitude of tasks in software engineering, such as technology transfer and software risk management. As initial task to be supported, we focus on the selection of technologies during the planning of software projects and improvement programmes. We claim that this task is the one where decision support based on TEBs occurs most frequently and where it has the most direct impact on software development.

The selection of software engineering technologies during the planning of software projects and improvement programmes should follow the paradigm of informed decision making. There is a large variety of factors affecting the "right" decision, and the result of the decision can be highly critical to the success of a software organisation. Hence, tool support is required for supporting human decision making rather than for prescribing an "optimal solution".

We base the decision support process implemented in *KONTEXT* on the principles of comprehensive reuse as introduced by Basili and Rombach [BR91]. This results in the following multi-staged decision support and selection process:

1 Determine the candidate processes for which a new technology can be applied, the product type to be developed, and the product quality goal to be achieved. – This information allows to pre-select a set of candidate technologies and their TEPs. From the context characteristics of these TEPs, a customised characterisation questionnaire can be constructed. It will be used to characterise the forthcoming project or improvement programme in a form that is suitable to conduct similarity-based retrieval of those TEPs that are most appropriate for the given reuse situation.

2 Characterise the forthcoming software project or improvement programme using the customised characterisation questionnaire. – Using this characterisation, a similarity-based ranking of the candidate TEPs can be conducted. This ranking is expected to support the final selection decision to be drawn by the human decision maker (i.e., the planner of the project or improvement paradigm).

3 Select the most appropriate technology (or technologies) for the forthcoming project or improvement programme. – This selection should be justified explicitly in order to facilitate the achievement of commitment and as a basis for later evaluation of the decision.

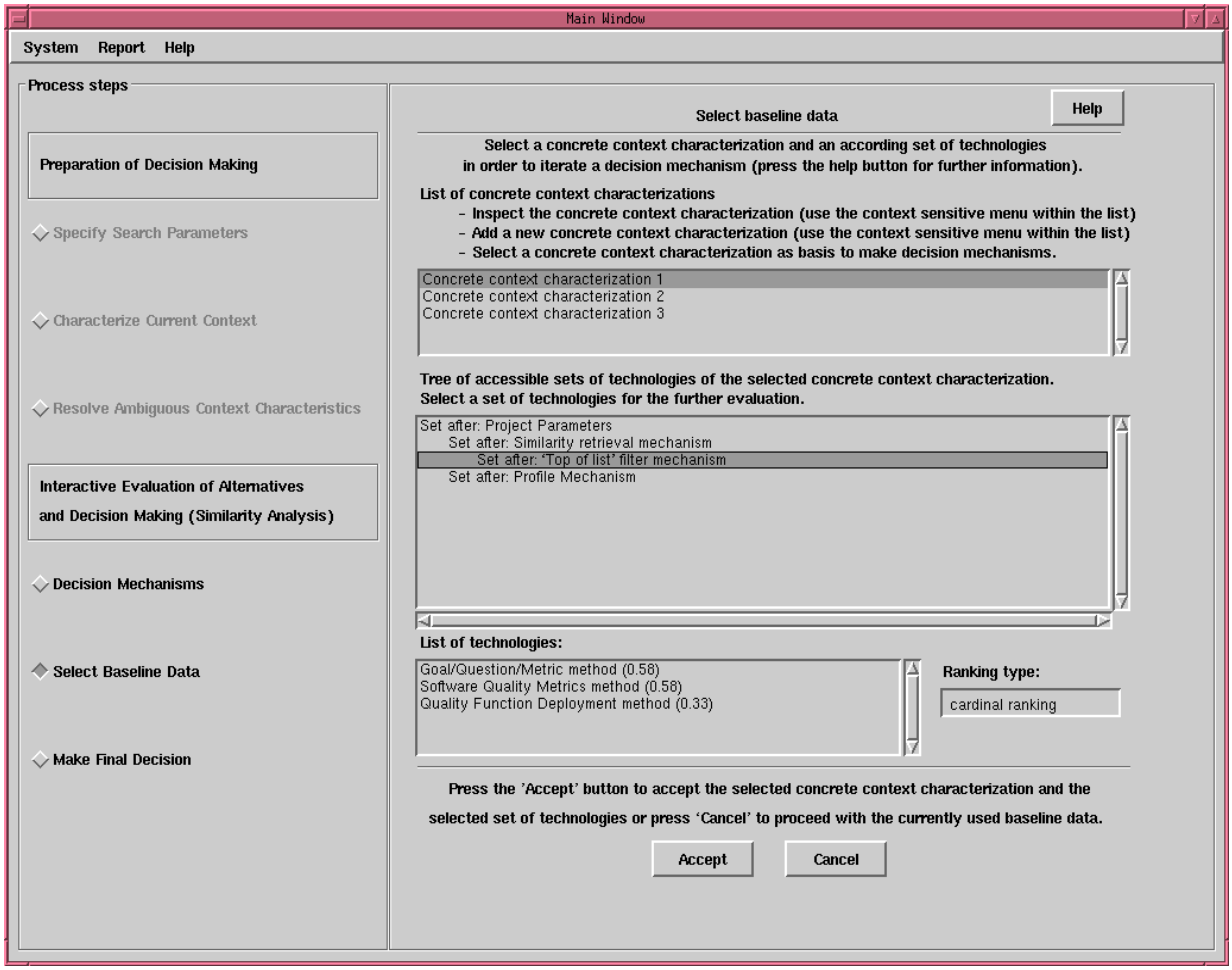

Figure 6: Selecting a technology

In *KONTEXT*, we have implemented several features that we consider beneficial for informed decision making in software engineering. Our pilot implementation demonstrates that these are practical solutions for advanced tool support for knowledge management in software engineering:

- Investigation of background information within the decision support process (e.g., technology definitions can be viewed before the final selection decision is drawn).
- Exploration of concurrent scenarios during the decision process. For instance, different characterisations of the forthcoming project can be used concurrently for similarity-based retrieval, and the results can be compared.
- Offering multiple different decision support mechanisms (e.g., multiple similarity functions or different multi-attribute decision making algorithms

[MPE96]; the decision maker can select the mechanism he/she regards most appropriate for the current type of selection).

- Backtracking and iteration of sub-steps of the decision process before the final decision is drawn. An example is the application of different decision support mechanisms using the same baseline information. Such a redundant decision process can increase the confidence in the results gained.
- Explicit justification of the final decision, in order to motivate the decision maker that he/she fully rationalises the decision.

The user interface of *KONTEXT* offers these features in its technology selection dialogue. It is shown in Figure 6. The rightmost column of the window guides the user through the multiple actions involved in technology selection. Among others, these are the preparation of the decision making by providing needed baseline information (e.g., characterisation of the forthcoming software project), the selection of a decision support method, and the drawing of the final decision and justifying it. The rightmost part of the window shows the dialogue for selecting baseline data for running a decision support algorithm that has been selected in a previous step. The baseline data involves one of possibly multiple alternative characterisations of the forthcoming project and a set of candidate TEPs. Since *KONTEXT* logs all activities of the decision process and their intermediate results, a hierarchy of gradually constrained sets of candidate technologies results from the iterative application of decision support algorithms. The second list in the window presents this hierarchical list of sets of candidate technologies and allows for selecting one of these sets. The technologies contained in the selected candidate set are depicted in the list at the bottom of the window.

For later evaluation of the application of the selected technology, the context characterisation and the entire trace of the decision support process are stored in the TEB. The following section explains how this information is used for refining and updating the TEB.

7 Empirical Evaluation of Technology Experience Packages

Objective of the empirical evaluation of technology experience packages is to possibly refine and update the contents of the TEB. When a technology is applied in a software project, this can be used as a case to investigate whether the information contained in a TEP is appropriate. The core question of empirical analysis is: Does the technology, when applied for the specified process and within the defined application context, really have an observable impact on the respective product quality? This question can be split into two separate investigations: (1) Has the product quality been achieved?, and (2) is the actual context situation of the project or improvement programme the same as defined in the TEP's context model and as it was expected when the technology was selected?

Depending on how the answers to these questions look like, there can be different consequences of empirical evaluation of TEPs:

- If the evaluation shows that the technology had significant impact on the product quality and that the actual application context was the same as in the TEP, then the contents of the TEB might be confirmed and provided with additional evidence. This would create deeper trust in the effectiveness of the technology when used later in similar application contexts.
- If the technology's impact on the product quality or the actual context characteristics were not as expected, then some kind of correction of the TEB is needed. Depending on the exact evaluation results and the contents of the TEB, the modifications can be quite different, ranging from small modifications of context models to the introduction of new variants of technologies and the partial re-design of multiple TEPs. For instance, it could turn out that a TEP on software inspections should better be split into two separate TEPs for variants of software inspections with and without inspection meeting. This would also require new, modified context models that clearly distinguish between the application contexts of the two inspections variants.

A detailed explanation of the analysis strategy for the empirical evaluation of TEPs would exceed the scope of this paper. Please refer to [BJvS99] for further details. The main objective is to identify a causal link between application of the technology and the achieved product quality under special consideration of the impact of the project context.

KONTEXT supports empirical evaluation by the following features:

- The trace of the technology selection process (cf. Section 6) including the initial characterisation of the project context (i.e., the expected characteristics

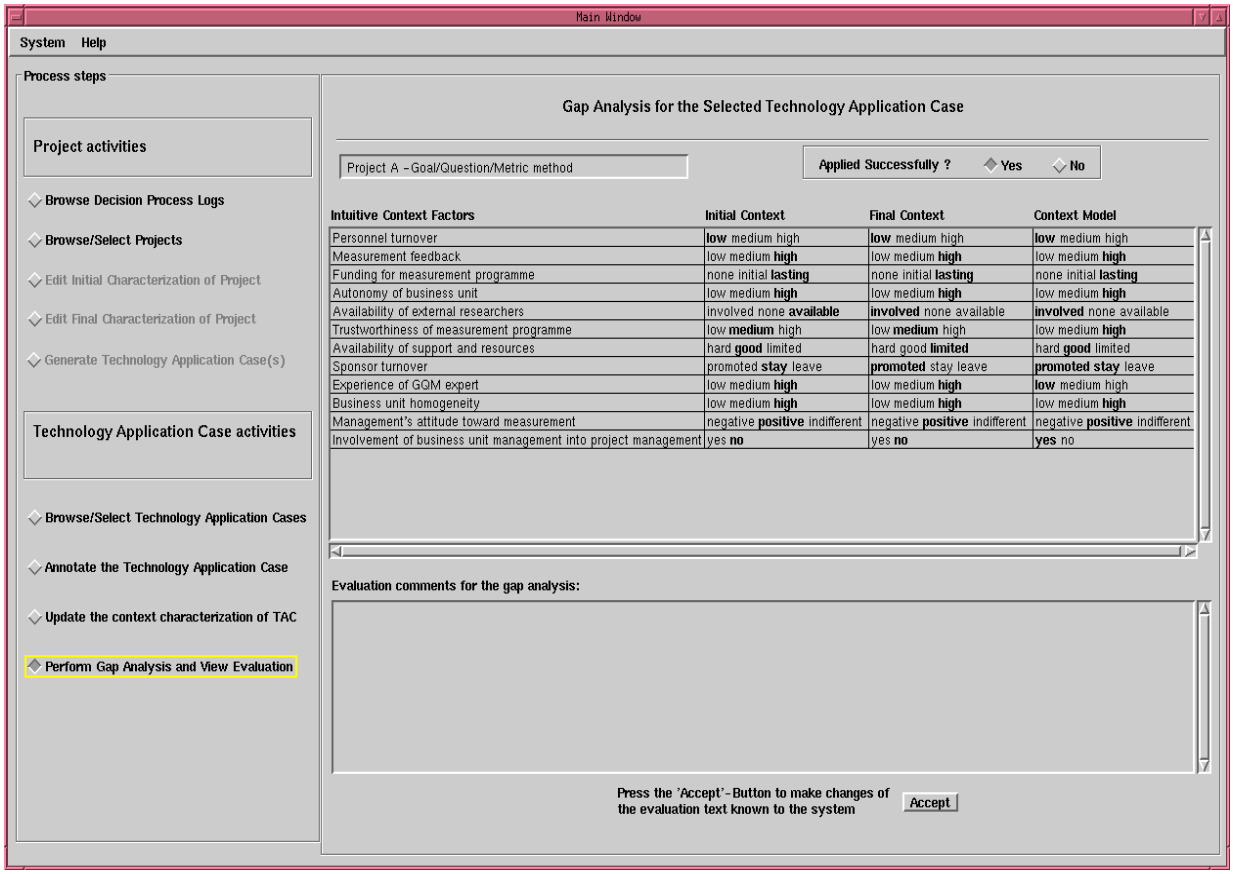

Figure 7: Empirical post-project analysis of a technology application case.

of the forthcoming project) are stored in the TEB as a basis for future evaluation.

- The actual context situation at the end of the project can be supplied. It is then stored in the TEB.
- *KONTEXT* offers a user dialogue that guides through the process of analysing the deviations between initially expected context characteristics, actual project characteristics, and the TEP's context model (see Figure 7).

The object model of *KONTEXT*'s TEBs (see Figure 8) contains objects for representing information about projects and about the application of selected technologies in these projects (*technology application case*). This way, *KONTEXT* integrates abstract information about reusable artefacts (i.e., the TEPs) with concrete information about the actual reuse of these artefacts. This is not only beneficial for empirical analysis of technology application. It also provides information relevant for informed decision making during technology selection.

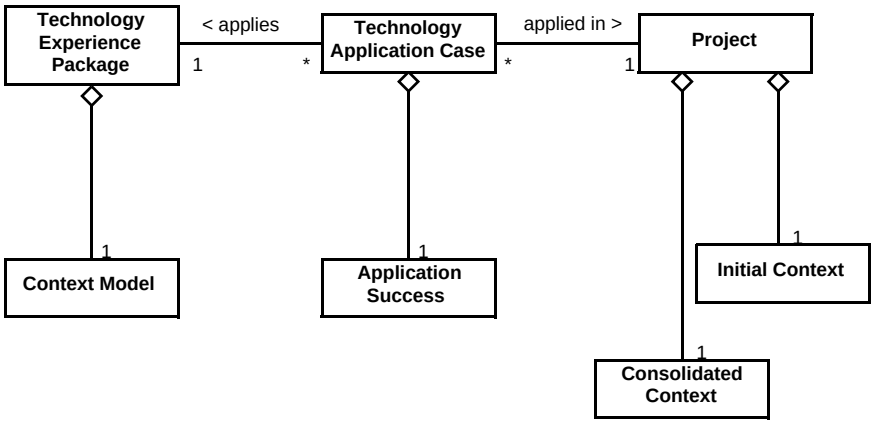

Figure 8: Technology Application Case, Project, and Technology Experience Package

8 An Example Technology Experience Base

A technology experience base of the presented kind is currently being developed in ESPRIT project PROFES[1]. The objective of PROFES is to support industries that have strong product-related quality requirements, such as the embedded systems industry, with an improvement methodology that focuses improvement actions on those elements of the software development process that contribute most to the critical product quality factors. It places emphasis on the continuous learning about the impacts that software engineering technologies and the processes in which they are applied have on product quality. This information is modelled in the form of so-called product/process dependencies (PPDs) and stored for reuse in a central repository.

The PPD repository is widely analogous to the technology experience base introduced above. PPDs deploy the representation scheme of TEPs. The usage processes on the PROFES PPD repository are designed in accordance with the presented TEP lifecycle model.

The PROFES PPD repository contains information about software engineering technologies that has been collected from multiple literature sources and from three industrial software organisations that have participated in the PROFES project. It will be offered for public use via the internet. So other software organisations can use the PROFES PPD repository to support the identification of improvement actions (i.e., the selection of technologies that are to be applied in forthcoming software projects) in their improvement programmes. They also will be able to feed back their experience and thus help to evolve and extend the repository.

First experience from the industrial applications of PROFES substantiate core components of the presented TEP lifecycle model: The TEP representation schema has proven appropriate for representing the information needed within technology experience bases, knowledge acquisition has shown effective for gaining TEPs, and the importance of product quality goals, processes, and context factors has been demonstrated (cf. [BJvS99]). Strategies for selection of technologies will be subject of further investigation during the PROFES-internal trials phase of the PPD repository. Future results will be reported in [PRO].

1 ESPRIT project 23239, PROFES (PROduct Focused improvement of Embedded software Processes); funded by the European Commission. http:/www.iese.fhg.de/Profes

9 Conclusions

The systematic management of knowledge about software engineering technologies and their application contexts is expected to reduce the risk of technologies-caused failure of software projects. Hence, knowledge management can be regarded an important means for further improvement of software engineering practices.

The Quality Improvement Paradigm (QIP) / Experience Factory (EF) approach ([BCR94]) provides appropriate solution concepts for realising customised knowledge management of experience on software engineering technologies that is packaged for reuse during the planning of software projects and improvement programmes. This experience is modelled explicitly in the form of so-called *technology experience packages* (TEPs) that are stored in repositories called *technology experience bases* (TEBs).

We have suggested a structure for organising TEBs and a comprehensive lifecycle model (1) for gaining TEPs through knowledge acquisition techniques, (2) for using them to support technology selection during the planning of software projects and improvement programmes, as well as (3) for updating and evolving TEPs based on the empirical evaluation of software projects.

Our approach differs from existing ones in that it addresses all three phases of the lifecycle model at a considerably detailed technical level. It provides guidance for organising and running technology experience bases. A prototypical tool called *KONTEXT* has been implemented in order to illustrate and substantiate the methodological concepts. An industrial trial application has provided first empirical evidence for the validity of the approach.

Acknowledgements

We want to thank our collegues at the SLI department of Fraunhofer IESE for the many valuable discussions and the feedback they have provided to our work. Markus Nick has provided comments on an earlier version of this paper.

References

[ADK98] Andreas Abecker, Stefan Decker, and Otto Kühn. Organizational memory. In "*Das aktuelle Schlagwort*" *im Informatik Spektrum*, volume 21 of *4*, pages 213–214. Springer Verlag, August 1998.

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

[Bir97] Andreas Birk. Modelling the application domains of software engineering technologies. Technical Report 014.97/E, Fraunhofer IESE, August 1997.

[BJvS99] Andreas Birk, Janne Järvinen, and Rini van Solingen. A validation approach for product-focused process improvement. Technical Report IESE-Report No. 005.99/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, 1999.

[BR91] Victor R. Basili and H. Dieter Rombach. Support for comprehensive reuse. *IEEE Software Engineering Journal*, 6(5):303–316, September 1991.

[BSA99] Andreas Birk, Dagmar Surmann, and Klaus-Dieter Althoff. Applications of knowledge acquisition in experimental software engineering. In *Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling, and Management (EKAW'99)*, Berlin, 1999. Springer. To be published.

[CDR95] CDR, Center for Design Research. http://gummo.stanford.edu/html/gcdk/dedal/index.html, 1995.

[CGvW98] R. Barcia C. Gresse von Wangenheim, A. von Wangenheim. Case-based reuse of software engineering measurement plans. In *Proc. of the 9th Int. Conference on Software Engineering and Knowledge Engineering (SEKE)*, 1998.

[CKO92] Bill Curtis, Marc I. Kellner, and Jim Over. Process modeling. *Communications of the ACM*, 35(9):75–90, September 1992.

[CY93] J. Conklin and E. Yourdon. Groupware for the new organization. *American Programmer*, sept 1993.

[Eri92] Henrik Eriksson. A survey of knowledge acquisition techniques and tools and their relationship to software engineering. *Journal of Systems and Software*, (19):97–107, 1992.

[FDES98] Dieter Fensel, Stefan Decker, Michael Erdmann, and Rudi Studer. Ontobroker: The very high idea. In *Proceedings of the 11th International Flairs Conference (FLAIRS-98)*, Sanibel Island, Florida, May 1998.

[GDS98] GDSS, Group Decision Support Systems. http://www.gdss.com/om.htm, 1998.

[Gla98] Robert Glass. *Software Runaways*. Prentice Hall, 1998.

[Hen97a] Scott Henninger. Capturing and formalizing best practices in a software development organization. In *Proc. of the 9th Int. Conference on Software Engineering and Knowledge Engineering (SEKE)*, 1997.

[Hen97b] Scott Henninger. Case-based knowledge management tools for software development. In *Automated Software Engineering: An International Journal*, volume 4. Kluwer Academic Publishers, 1997.

[IBM98] IBM. http://www.software.ibm.com./data/ids/, 1998.

[KPM95] KPMG Ltd. Runaway projects–cause and effects. *Software World*, 26(3), 1995.

[Krö98] Felix Kröschel. A system for knowledge management of best software engineering practice. Master's thesis, University of Kaiserslautern, Kaiserslautern, Germany, November 1998.

[Lot] Lotus. http://www.lotus.com.

[MPE96] Mansooreh Mollaghasemi and Julia Pet-Edwards. *Making Multiple-Objective Decisions*. IEEE Comuter Society Press, 1996.

[MT90] Ackerman M.S. and Malone T.W. Answergarden: A tool for growing organizational memory. In *Proc. of the ACM Conference on Office Information Systems*, pages 31–39, 1990.

[O'L98] Daniel O'Leary. Using ai in knowledge management: Knowledge bases and ontologies. *IEEE Intelligent Systems*, pages 34–39, May/June 1998.

[Ope] Open Text. http://www.opentext.com.

[PRO] PROFES. ESPRIT project 23239 (Product-FOcused improvement of Embedded Software processes). http://www.ele.vtt.fi/profes/.

[Rei97] Ulrich Reimer. Knowledge acquisition for content selection. In *21st Annual German Conference on AI '97*, Freiburg, September 1997. http://www.dfki.uni-kl.de/km/ws-ki-97.html.

[RV95] H. Dieter Rombach and Martin Verlage. Directions in software process research. In Marvin V. Zelkowitz, editor, *Advances in Computers, vol. 41*, pages 1–63. Academic Press, 1995.

[RZ93] Li N. R. and M. V. Zelkowitz. An information model for use in software management estimation and prediction. In *Second International Conference on Information and Knowledge Management*, pages 481–489, Washington, DC, November 1993.

[Sen90] Peter M. Senge. *The Fifth Discipline. The Art and Practice of The Learning Organization*. Bantam Doubleday Dell Publishing Group, Inc., New York, 1990.

[Shu97] S. Buckingham Shum. Negotiating the construction and reconstruction of organisational memories. *Journal of Universal Computer Science*, 3(8):899–928, 1997.

[SKR+94] Peter M. Senge, Art Kleiner, Richard B. Ross, Bryan J. Smith, and Charlotte Roberts. *The Fifth Discipline Fieldbook. Strategies and Tools for Building a Learning Organization*. Bantam Doubleday Dell Publishing Group, Inc., New York, 1994.

[Sta95] Standish Group. CHAOS, 1995. www.standishgroup.com/chaos.html.

[SWI97] Department of Social Science Informatics Sociaal Wetenschappelijke Informatica. http://www.swi.psy.uva.nl/projects/newkactus/reports.html, 1997.

[TA97] Carsten Tautz and Klaus-Dieter Althoff. Using case-based reasoning for reusing software knowledge. In D. Leake and E. Plaza, editors, *Proceedings of the Second International Conference on Case-Based Reasoning*. Springer Verlag, 1997.

[TZ98] Roseanne Tesoriero and Marvin Zelkowitz. A web-based tool for data analysis and presentation. *IEEE Internet Computing*, 2(5):63–69, 1998.

[Wii95] Karl Wiig. Knowledge management methods. practical approaches to managing knowledge. In *Knowledge Management: The Central Management Focus for Intelligent-Acting Organizations*, volume 3. Schema Press, Ltd, 1995.

[You93] E. Yourdon. *Decline & Fall of the American Programmer*. Computing. Yourdon Press, 1993.

Document Information

Title:	A Knowledge Management Lifecycle for Experience Packages on Software Engineering Technologies
Date:	February 1999
Report:	IESE-007.99/E
Status:	Final
Distribution:	Public