Applications of Knowledge Acquisition in Experimental Software Engineering

Andreas Birk, Dagmar Surmann, and Klaus-Dieter Althoff

Fraunhofer Institute for Experimental Software Engineering (Fraunhofer IESE),
Sauerwiesen 6, D-67661 Kaiserslautern, Germany
{birk,surmann,althoff}@iese.fhg.de

Abstract. Many tasks in experimental software engineering (ESE) involve the acquisition of knowledge. Only for very few of them systematic knowledge acquisition (KA) practices have been established. It is expected that these ESE tasks can be accomplished more effectively if the application of appropriate systematic KA methods is fostered.
Most reports on KA applications in software engineering address only some selected aspects. A broader ESE perspective with its additional facets (e.g., quality and knowledge management issues) has not yet been presented so far.
This paper surveys applications of knowledge acquisition in experimental software engineering, introduces a repository of knowledge elicitation (KEL) techniques, and suggests a methodology for the development of customised KA methods in experimental software engineering. Repository and methodology aim at fostering the dissemination of systematic KA practices in ESE. They are applied at Fraunhofer IESE to develop methods for the acquisition of experiential software engineering knowledge.

Keywords: knowledge acquisition, knowledge management, experimental software engineering

1 Introduction

Software engineering (SE) involves a multitude of knowledge-intensive tasks: Elicitation of user requirements for new software systems, identification of best software development practice, experience collection about project planning and risk management, and many others.
In addition, the discipline of experimental software engineering (ESE)[1] places particular emphasis on knowledge management and knowledge-based support. It builds on the assumption that continuous learning and systematic reuse of learnt

[1] Experimental Software Engineering [1] covers all traditional fields of Software Engineering. It places particular focus on the empirical investigation of Software Engineering concepts such as techniques, methods, or tools. Approaches for managing the gained empirical knowledge play an important role in ESE.

knowledge and experience are crucial for the further development of today's software development and management practices.
Despite the importance of knowledge management in ESE, organised and mature approaches to knowledge acquisition (KA) are rare. Compared to other areas of engineering, SE is still quite young. Technological developments are progressing fast. For these reasons, the often needed KA and knowledge management components of methods and techniques have not yet been developed very far. However, for the same reasons, effective knowledge acquisition and knowledge management (KM) become more and more important for sustained business success.
The objective of our work is to foster the further dissemination and use of advanced KA methods in ESE. This paper presents the approach through which we want to achieve this objective. The approach consists of three elements:

- Survey and characterisation of KA applications in ESE
- A repository of reusable knowledge elicitation (KEL) techniques and experience
- A methodology for developing customised KA methods for specific ESE tasks

The survey of KA applications provides an overview of the various tasks in experimental software engineering that can benefit from advanced and systematic knowledge acquisition practices. The characterisation of these applications illustrates the need for customised KA methods. It also is the first step to identify requirements for the development of such customised methods.
A repository of knowledge elicitation techniques is important for disseminating good KA practice throughout experimental software engineering. Many KEL techniques that could be beneficial to ESE are just not known to software engineers. Operational definitions of these techniques need to be collected and made accessible to software engineers. They must be supplied with experience about when and how to use which technique. This information facilitates the selection of KEL techniques for a specific knowledge acquisition task and supports the application of the KEL techniques.
Such a repository does not only support knowledge acquisition and knowledge management in ESE. It also is a knowledge management application itself, because the repository needs to be kept alive. It must be extended continuously, and experience from the application of KEL techniques should be fed back into it.
A methodology is needed to guide software engineers in the development of customised KA methods. Such a methodology starts with the characterisation of a knowledge acquisition task and selects appropriate KEL techniques from the repository. The KEL techniques must then be integrated with each other and supported by appropriate tools. The result is an operational, customised KA method for the specific application task.
This paper is structured according to the three elements of our approach: Section 2 surveys applications of knowledge acquisition in experimental software engineering, Section 3 presents the repository of KEL techniques, and Section 4

describes the knowledge acquisition methodology. A discussion and an outlook on future work are addressed in Section 5. The remainder of this first section briefly introduces the fields of experimental software engineering and knowledge acquisition. It also lists requirements on KA applications in ESE, which were used to guide the development of the approach presented.

1.1 Experimental Software Engineering

Software engineering aims at providing technologies that can be used for developing software and for managing software development. This involves the definition, selection, tailoring, and integration of principles, methods, techniques, and tools in order to achieve a software product that meets the desired quality, time, and cost requirements. For managing these requirements and for demonstrating that they actually have been achieved, analytical and empirical measures must be applied [2] [1].
Experimental software engineering is a branch of software engineering that addresses problems and research questions of software development and improvement through an experimental approach. It utilises systematically designed experiments and other kinds of empirical studies in order to enhance available knowledge about the software domain. ESE is built on the principle of continuous learning and reuse of experience, which is defined through the Quality Improvement Paradigm / Experience Factory (QIP/EF).

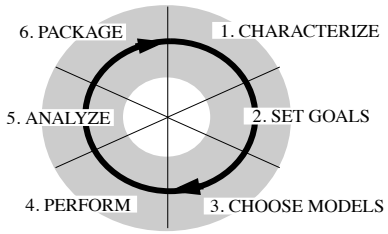

Fig. 1. The Quality Improvement Paradigm

The QIP is a six-step proceeding for structuring software development and improvement activities (see Figure 1). It involves three overall phases: planning, execution, and evaluation of the task. The planning phase is based on the explicit characterisation (QIP1) of the initial situation, the identification of the goals to be achieved (QIP2), and the actual development of the plan (QIP3). The plan then guides the systematic execution of the task (QIP4). The subsequent evaluation phase involves the analysis of the performed actions (QIP5) and the packaging of the lessons learnt into reusable artifacts (QIP6). The evaluation allows to learn for similar tasks in the future.

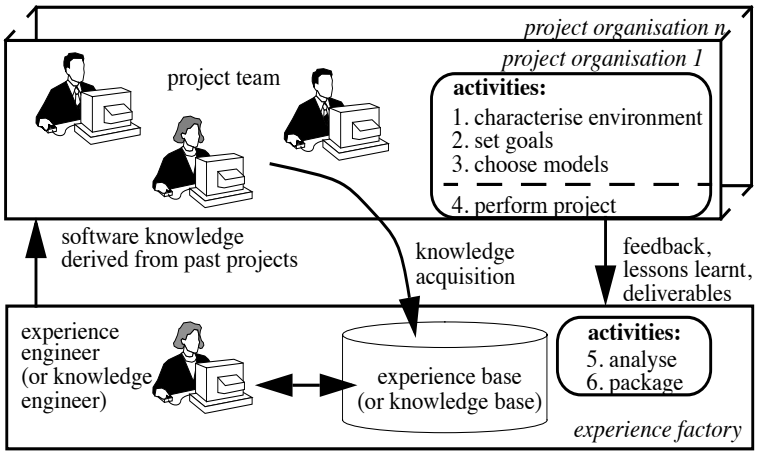

Fig. 2. The Experience Factory

The experience factory is a logical and/or physical organisation that supports project developments by analysing and synthesising all kinds of experience, acting as a repository for such experience, and supplying that experience to various projects on demand. The experience factory complements the project organisation [2]. The experience factory is mainly responsible for conducting steps 5 and 6 of the QIP while steps 1 to 4 mainly concern the project organisation (see Figure 2).
ESE has many relations to knowledge engineering and knowledge management. KA from experienced software professionals is an important means for gaining the knowledge and insight to answer ESE questions.

1.2 Knowledge Acquisition

Knowledge acquisition (KA) is the transfer and transformation of expertise from some knowledge source to some explicit knowledge representation-usually denoted as knowledge base-that enables the effective use of the knowledge. This definition is based on the one by Hayes-Roth et al. from 1983 [3]. It has been generalised slightly to meet the application of knowledge acquisition in experimental software engineering as addressed in the remainder of this paper.
A KA method is an organised approach to knowledge acquisition. It involves a defined process and guidelines for process execution. A knowledge acquisition methodology defines and guides the design of KA methods for particular application purposes. Section 4 suggests a KA methodology for experimental software engineering. Knowledge elicitation (KEL) denotes the initial steps of knowledge acquisition that identify or isolate and record the relevant expertise using one or multiple KEL techniques. A KA method can involve a combination of KEL techniques which is then called KEL strategy. These terms are used differently

by different authors. We have chosen our definitions to meet the specific terminology needs of this paper.
Musen [4] lists several characteristics of knowledge acquisition that need to be considered when applying KA methods:

- Knowledge acquisition is a process of joint model building. A model of expertise is built in co-operation between a domain expert (i.e., the knowledge source) and a knowledge engineer.
- Much knowledge is tacit (i.e., it is not directly accessible). Appropriate KEL techniques are needed to make it explicit.
- The results of knowledge acquisition depend on the degree to which the knowledge engineer is familiar with (a) the domain of the knowledge to be acquired, and (b) its later application.
- The results of knowledge acquisition depend on the formalism that is used to represent the knowledge. Knowledge acquisition is most effective if knowledge representation is epistemologically adequate (i.e., all relevant aspects of expertise can be expressed) and usable (i.e., suits all later usage needs).

These characteristics of knowledge acquisition provide guidance for the design of KA methods. For example, they imply that KA methods must assure that the knowledge engineer becomes familiar with the application domain. In Section 4 a knowledge acquisition method is presented that reflects these characteristics.

1.3 Requirements on KA in ESE

Experimental software engineering puts specific requirements on knowledge acquisition that are quite different from KA applications in other engineering sciences or from the development of knowledge-based systems (KBS):

- For most KA applications in ESE, the required knowledge exists only implicitly, codified, and informally. Hence, it can be quite difficult to access the knowledge. This imposes high requirements on the validity of the KEL techniques and strategies used.
- The results of KA in ESE are often the basis for further technical or managerial activities. These activities depend very much on the reliability of KA results. For this reason, the validity of KA methods and their results is an important concern in ESE.
- ESE involves a wide variety of target knowledge representation formats for KA, which can be highly specific to ESE. Many of these knowledge representation (KR) formats are different from the traditional rule-, frame-, or case-based formalisms involved in KBS. For this reason, the target KR format becomes an important criterion to assess the appropriateness of a candidate KEL technique.
- In addition, an ESE artefact (e.g., a design document) often involves multiple different knowledge types (e.g., facts as well as rules and policies). Hence, an appropriate KA method might require a combination of multiple KEL techniques.

- In ESE, KA is not only useful for providing problem-solving support in the form of knowledge-based systems. It can also provide the knowledge to effectively perform technical and managerial tasks, and it can provide input to corporate knowledge management systems and improvement programmes.
- Knowledge-based systems have not yet found their way to wide-spread use in the software industry (cf. [5] [6]). To foster their further dissemination, a gradual transition from current ESE practices via knowledge management implementation to KBS is recommended. This is an additional challenge for KA in ESE, asking for KA methods that adapt to specific needs of KM applications.
- Systematic knowledge acquisition should be integrated into many ESE tasks. As a consequence, KA methods should be applied also by non-knowledge engineers. This requires a strong need for operational definitions of KA methods and guidance for their application.

A KA methodology for ESE must take these requirements into consideration and provide a well-organised approach to the selection of appropriate KEL techniques. The KA methodology suggested in Section 4 reflects these requirements.

2 A Survey of KA Applications in ESE

Experimental software engineering involves a multitude of knowledge acquisition tasks. They can differ in quite a variety of aspects such as involved knowledge types, knowledge sources, knowledge users, modes of knowledge use, and target knowledge representation. A good starting point to provide tailored and effective methodological support to these knowledge acquisition tasks in ESE is to survey these tasks systematically, and to characterise them appropriately.
This section presents such a survey and characterisation of KA applications in experimental software engineering. It thus demonstrates the wide variety of KA-related tasks, which has often not been recognised sufficiently by ESE. For the field of knowledge acquisition, the survey provides a map of possible applications for established KA approaches. The survey is not meant to be comprehensive and complete. It was our intention to build on previous work in SE and to complement it with a specific ESE perspective.

2.1 Literature Review

Applications of knowledge acquisition in software engineering have been described-among others-by [6] [7] [8] [9]. A particular tradition in systematic KA can be found in the field of requirements engineering (RE) (cf. [10] [11] [12] [13]).
Eriksson [7] lists three major application areas of knowledge acquisition in software engineering: Initial feasibility studies, requirements specification, and the identification of solution approaches for design and implementation problems. He summarises that KA is a broad activity which may be useful at many stages in the software development process. Some KA methods were already used in SE, albeit often in a less explicit and systematic manner than in KBS development.

Grogono [6] addresses the mutual interrelations between expert systems and software engineering. Thus, indirectly, he also covers the need for knowledge acquisition in SE. The expert systems that he lists require the following kinds of knowledge: Requirements specifications, expertise on design structures (i.e., products or artifacts), expertise on the design and implementation processes (mainly in the form of rules or heuristics), as well as software process models (i.e., procedure-like representations).
Briand et al. [8] describe a method for estimating software development effort. It is a hybrid approach that combines acquisition of experiential knowledge with empirical data from past projects. The acquired causal models allow for effort predictions that are based on significantly less empirical data than would be needed otherwise. Wilson and Hall [9] use construct elicitation to investigate the perceptions of software quality that they found in a number of IT organisations. The various analogies between knowledge engineering and requirements engineering are surveyed and investigated by Angele and Studer [10] and by Shaw and Gaines [11]. A recent publication by Weidenhaupt et al. [12] surveys and investigates the application of various scenario techniques for the acquisition of system requirements. Maiden and Ncube [13] describe an approach to the acquisition of requirements for the selection of commercial off-the-shelf software that involves multiple semi-structured interview techniques.

2.2 Survey Results

Most reports on KA applications related to software engineering either address only some particular aspects (e.g., requirements elicitation or cost estimation) or focus on software development tasks only. A broader ESE perspective with its additional facets (e.g., quality, improvement, and knowledge management issues) has not yet been presented so far. To gain such a broader perspective on KA applications in experimental software engineering as a basis for our further work, we have investigated multiple literature sources and interviewed SE experts. The results are shown in Table 1. The table demonstrates the wide area of applications of KA in ESE. Each application is characterised briefly.
Starting point of our investigation was a taxonomy of ESE tasks. It is listed in the leftmost column of Table 1. The tasks are grouped into three categories: Product engineering (i.e., the typical software development tasks), management (e.g., project or quality management), and support (i.e., all activities that are not directly related to the product development but that ease, facilitate, and improve it). For each task, a set of knowledge items (i.e., artefacts or concepts) was identified, which are usually gained within the related task by some kind of knowledge acquisition activity. Each knowledge item is characterised using its subject topic and the typical kinds of knowledge encoding (i.e., whether it is contained in documents, available in the minds of humans, or present in the form of processes and procedures).
Each set of knowledge items is supplied with information about the typical knowledge sources (e.g., a SE role or a document), its knowledge users, and the target knowledge representations in which it is documented and used once

ESE Task	Knowledge Items: Subject Topics	Knowledge Items: Encoding	Knowledge Sources	Knowledge Users	Target Knowledge Representations
Product Engineering					
Requirements Analysis	Requirements, Business Processes, Use Cases	H, P, D	U, C, M, SE, PD, S	SE	S, G, U, F
Architecture Design	Reusable Artefacts, Templates, Patterns[1]	D, H	PD, L, SE, S	SE	G, S, F, U
Detailed Design					G, S, F, U
Implementation					F, G
Integration and Testing	Test Plans, Test Cases	D, H, P	U, C, SE, QM, SP	SE	F, G, U, S
Maintenance	Programme Understanding	D, H, P	SE, S, PD	SE	F, G
Software Acquisition	Requirements, Third-Party Products	D, H	PD, L, SE	SE	S, U, G
Management					
Project Management	Time and Effort Estimates, Schedules, Staffing Plans	H, P, D	PM, PD, L, SP	PM	F, S, G
Quality Management	Quality Plans	D, H, P	PD, PM, QM, SP, L	QM	S, G
Risk Management	Risk Mitigation Plans	H, P, D	PM, PD, SP, L	PM	S
Support					
Configuration Management	Configuration Management Plans	D, H, P	PD, SE, QM, SP	SE, QM, PM	F, G, S
Documentation	User Documentation	D, H, P	PD, SE, U, S	TW	S, G
Process Modelling	Process Models	H, D, P	SE, PM, QM, PD, SP	SE, QM, PM	F, G, S, U
Process Enactment	Processes, Guidelines	H, P, D	SE, PM, QM, SP, PD	SE, QM, PM	S, G
Process Automation[2]	Process Models, Configuration Management Plans, System Architecture, Code[3]	D, P, H	PD, SP, SE, PM, QM	SE, QM	F, G, S
Process Assessment	Various aspects of the software engineering practices	P, D, H	SP, PD, SE, PM, QM	PM, QM	S, G, U
Measurement	Quality Goals, Understanding and Definition of Qualities, Products, and Processes, Expected Phenomena, Measurement Plans, Interpretations of Observed Phenomena	D, P, H	PD, SP, SE, PM, QM	PM, QM, SE	F, G, S
Improvement	Improvement Plans	D, H, P	PD, SE, PM, QM, SP	QM, PM	S, G

[1] Typical subjects of these knowledge types are architectures, data models, algorithms, and code.

[2] Note: This involves process automation using CASE tools, and process support using SE environments.

[3] These items are structure- and process-related knowledge about different software engineering artefacts or concepts relevant for setting-up process automation.

Legend: These lists are sorted by relevance. Most relevant items appear first.
Knowledge Types (Encoding): D=Documents, H=Humans, P=Processes.
Knowledge Sources and Knowledge Users: U=User, C=Customer, M=Marketing, PM=Project Management, QM=Quality Management, SE=Software Engineer, TW=Technical Writer, S=Existing System, PD=Existing Project Documentation, L=Literature, SP=Software Process.
Target Knowledge Representation: G=Graphics, U=Unstructured and S=Structured or Semi-Formal Natural Language, F=Formal Language.

Table 1. Overview of KA applications in ESE

it has been acquired. For each set of characteristics, the order in which they are listed indicates an order of relevance.

2.3 Observations from the Survey

The survey and the characterisation of knowledge acquisition applications in experimental software engineering provide an interesting perspective on the detailed, specific KA requirements of ESE. Observations can be made that are useful for selecting KEL techniques and for developing KA methods for application in experimental software engineering. In the following, the most relevant observations are summarised.

Product Knowledge vs. Process Knowledge. Two kinds of knowledge are most important for the gross number of ESE tasks: Product knowledge and process knowledge. Product knowledge addresses structure and other characteristics of artefacts (e.g., system architecture or functionality). Process knowledge deals with how ESE tasks should be performed (e.g., the development process, policies, and guidelines), and how the tasks interact.

Dependency Chains between ESE Tasks. For product engineering and support tasks, especially, it is typical that there are chains of dependent tasks: Design tasks depend on requirements analysis, and implementation depends on design. Likewise, process assessment and measurement depend on process modelling, and improvement depends on process assessment and measurement.

These dependency chains have implications on knowledge encoding: Knowledge that is needed to accomplish the tasks in the chain can mainly be acquired from human experts, especially the first tasks. During KA such knowledge is represented explicitly, so that the subsequent tasks can widely rely on documented knowledge.

High Variety of Knowledge Sources. The lists of knowledge sources for the different ESE tasks can be quite long. The knowledge sources of a certain task can be quite different. This implies that the KEL techniques used to gain the required knowledge must also be quite different, because different groups of persons (e.g., software engineers vs. customers) can show very different communication styles and terminologies.

Support Tasks Have Many Different Knowledge Users. By definition of the taxonomy that is used to structure the ESE tasks, the knowledge users of product engineering tasks are software engineers, and those of management tasks are project or quality managers. In contrast, each support task can have multiple knowledge users, and the sets of users for two support tasks can be different.

The survey of KA applications in experimental software engineering and their characterisations provide a starting point for the development of customised KA methods that can be used to accomplish these tasks. The two following sections suggest an approach by which the development of such methods can be supported.

3 An Experience Base of Knowledge Elicitation Techniques

Software engineering can benefit from the adoption of advanced KA practices. Therefore, a body of knowledge about KA needs to be collected, made accessible, and disseminated to software engineering professionals. Experience about KEL techniques is of particular interest to experimental software engineering, because these are the basic elements needed to develop customised KA methods for the various ESE tasks.
This section presents the repository (or experience base) of KEL techniques that has been built at Fraunhofer IESE. We describe the structure of the chosen knowledge representation and outline how the knowledge was collected and how it is used.
To support systematic KEL practices in ESE, the following information needs to be provided to software engineering personnel:

- Concise and operational definitions of the KEL techniques.
- Information about the application context of the KEL techniques (i.e., in which situations it can be applied, and in which situations it is inappropriate).
- Traceability information and literature references that allow to access further information about the KEL techniques.

The representation structure we have chosen to describe experiences about KEL techniques (in the following denoted as experience packages) meets these requirements. Table 2 shows an example experience package. It has the form of a table with pre-defined information blocks. The upper part of the table contains the definition and classification of the technique as well as references and traceability information (i.e., its name, and slots for sources, classification, relationships, description, and characteristics). The lower part contains information relevant to selecting and using the technique (i.e., its application context). Its slots are: Prerequisites, advantages, disadvantages, risks, and notes.
Each entry that has been acquired from some literature source is supplied with a reference to this source. Information about application context is classified using keywords at the beginning of the statement. The classes indicate aspects of knowledge elicitation to which the statements refer. For instance, KTYP stands for knowledge types and marks statements like "Is appropriate for eliciting facts, conceptual structure, causal knowledge, and justification" in the advantages slot of Table 2. ELIC and EXPT mark statements that refer to the roles of elicitor and expert, respectively.
The experience base currently contains about 30 experience packages. Focus is put on KEL techniques for elicitation from individual human experts in interview-like sessions. Examples are semi-structured interview, retrospective case description, list-related tasks, teachback, construct elicitation using repertory grids, and laddering.
The information contained in the experience packages has been gained from multiple literature sources that survey KEL techniques or report on experiences

from using some of them (cf. [14] [15] [16] [17]). The raw information that has been found in these texts has been categorised and structured gradually to gain the experience packages. However, once the experience package structure had been established, it was quite straight-forward to add new KEL techniques or to extend the information about already catalogued ones.

Semi-Structured Interview	
Sources	Cordingly [14], Cooke [15], Welbank [16]
Classification	Interview/Semi-structured Interviews [14] Interviews/Structured Interviews [15]
Relationships	Kind of Interview
Description	The interviewer has a list of prepared questions. But the order in which they are covered and the words used to express them may vary from interview to interview. Many of the questions are open questions. [14], [15]
Characteristics	• Puts more demands on the interviewer than do fully structured and pre-determined interviews. [14]
Prerequisites	• Preparation of generic questions and coarse outline of interview structure. • Some basic familiarity of the elicitor with the domain and the tasks for which knowledge needs to be acquired.
Advantages	**CNTS** Structure provides more systematic and complete coverage of the domain than unstructured interviews. [15] **STYL EXPT ELIC** Structure tends to be more comfortable for both expert and elicitor. [15] **PERF STYL** The interview can flow smoothly. [14] **CNTS** The interviewee's associations between topics can be identified, because he/she has the freedom to follow spontaneous associations during the interview. [14] **KTYP** Is appropriate for eliciting facts, conceptual structure, causal knowledge, and justification. [16]
Disadvantages	**PROC ELIC** Requires more preparation time and domain knowledge than unstructured interview. [15] **KTYP** Is inappropriate for eliciting expert's strategy. [16]
Risks	**KTYP** Can be inappropriate when used to elicit rules, weight of evidence, and context of rules. [16]
Notes	• Answers to one question may arise as part of answers in another question. [14] • The wording of questions can be adopted to the vocabulary of the interviewee. [14]

Table 2. Example experience package for a KEL technique.

The experience packages can be used by software engineering professionals in multiple ways to gain an overview over KEL techniques, and to select some that meet the requirements of the tasks they have to accomplish. The structure of

the experience packages and the keywords that classify each statement allow to search or browse the information for various aspects or subject topics. A KA method that illustrates very well how individual KEL techniques can be integrated in a systematic manner has been presented by Briand et al. [8].
The experience base is extended and updated gradually. Currently, it is provided as versioned electronic documents with some basic hypertext functionality to access structure elements or indexed parts of experience packages. Our future plans are to transfer them into HTML format and offer them as an on-line experience base through the intranet. In addition, we are about to implement the experience base in a prototype knowledge management system that is specialised toward decision support for the selection of SE technologies during project planning. This knowledge management system called *KONTEXT* [2] [18] [19] supports the entire life-cycle of *Technology Experience Packages (TEPs)*, involving (1) knowledge acquisition and modelling of TEPs, (2) decision support for the selection of software engineering technologies, and (3) empirical evaluation of technology application and update of TEPs. *KONTEXT* is a research prototype that is being applied for internal uses at Fraunhofer IESE.
Currently, works are under way to build two variants of *KONTEXT*: One is for offering *Technology Experience Bases* and the TEPs they contain over the internet. The other implements the core functions and data model of *KONTEXT* into Fraunhofer IESE's corporate information network infrastructure. As soon as these works will be completed, the experience base on KEL techniques can be offered as part of a comprehensive knowledge management system that–among other features– allows for knowledge annotation and continuous knowledge evolution of KEL experience.
The knowledge representation of *KONTEXT* in the Fraunhofer IESE corporate information network deploys an object-oriented formalism and associated case-based knowledge representation. Based on a collaboration with a commercial case-based reasoning (CBR) tool provider, a CBR tool has been developed for supporting the retrieval of experience packages [20] [21] [22]. For a public case base a first implementation of the CBR tool has already been validated empirically [23][3]. Further experimental validation of *KONTEXT* is currently being prepared.

4 A KA methodology for ESE

The further dissemination and implementation of systematic knowledge acquisition practices in experimental software engineering requires an appropriate methodological framework. This section suggests a methodology for guiding the development and application of customised KA methods in ESE. The methodology starts with a characterisation of KA tasks (cf. Table 1) as starting point and selects appropriate KEL techniques from the experience base (cf. Section

[2] *KnOwledge maNagement base on the application conTEXt of software engineering Technologies.*
[3] http://demolab.iese.fhg.de:8080/

3). It thus integrates the two other elements of our approach that have been introduced in the previous sections.

Table 3 depicts the structure of the methodology. It involves four phases and twelve steps. The initial phase is a pre-study for gaining background information and requirements on design and application of the KA method. The second phase, KEL strategy development, is the core part of the methodology. It defines the KEL strategy. The two subsequent phases are knowledge elicitation and modelling. Hence they address the application of the KA method.

The KA methodology is described in detail in [24]. The following sub-sections outline its overall structure.

Phase	**Step / Sub-Step**
Pre-Study	Conduct pre-study on subject topic
	Conduct pre-study on usage processes
	Identify knowledge representation
KEL Strategy Development	Identify requirements and candidate KEL techniques • Characterise application situation • Identify applicable KEL technique • Identify further pre-study needs
	Define KEL strategy • Select KEL techniques • Integrate KEL techniques • Document KEL strategy
	Develop support tools and validate KEL strategy • Identify requirements on KEL execution • Develop support tools • Validate KEL strategy and support tools
Knowledge-Elicitation	Plan knowledge elicitation
	Prepare knowledge elicitation
	Conduct knowledge elicitation
Knowledge-Modelling	Construct knowledge model
	Validate knowledge model
	Release knowledge model

Table 3. Overview of the KA methodology.

4.1 Pre-Study

The first phase of the methodology aim at making the knowledge engineer familiar with the subject topic and the usage processes of the knowledge models to be acquired. Pre-study of the subject topic involves an investigation of relevant and available knowledge sources. Based on this information, a suitable knowledge representation formalism is determined or designed from scratch.

4.2 KEL Strategy Development

The core phase of the methodology is the actual design of the KEL strategy. It starts with identification of requirements and candidate KEL techniques. If further pre-study is needed, exploratory knowledge elicitation activities should be planned explicitly. The identification of appropriate techniques can be supported by an experience base of KEL techniques (cf. Section 3). The individual techniques identified must then be integrated, and the method must be defined. Finally, support tools for knowledge elicitation must be provided, and the KEL strategy and its support tools must be validated.
Explicit and operational definition of KA methods is recommended, because it eases the dissemination and re-use of the methods. It also facilitates the planning of KA activities in the context of software projects, which have often tight schedules. Furthermore, in experimental software engineering the KA methods might be applied by persons with little experience in knowledge acquisition. So operational methods can provide beneficial guidance and support.

4.3 Knowledge Elicitation

The actual execution of a customised KA method starts with knowledge elicitation. Knowledge elicitation activities must be planned in accordance with the schedule of the software projects in which the experts are working, and by which the knowledge will be used later. Preparation activities involve customisation of questionnaires and providing the technical infrastructure for knowledge elicitation. During knowledge elicitation sessions, notes or records need to be taken. Possibly some intermediate or mediating knowledge model is being developed.

4.4 Knowledge Modelling

The knowledge modelling phase translates the KEL results into an appropriate knowledge model. The model needs to be validated thoroughly, because many ESE tasks may build on it. Finally, the validated knowledge model is released and disseminated. In some cases release and dissemination of KA results can become a major task. For instance, the dissemination of acquired good design practices may require an entire training programme.

4.5 Example KA Method

The methodology can be illustrated by a method for software development effort estimation called COBRA (COst estimation, Benchmarking, and Risk Assessment) [8]. COBRA applies several KEL techniques in an integrated and very systematic way. It has an operational definition and is based on explicit rationales. COBRA has been applied successfully in industrial environments. Thus it supports the appropriateness and validity of the KA methodology. Details about COBRA and its relation to the presented KA methodology are addressed in [24].

4.6 Validation of the KA Methodology

The KA methodology for ESE needs to be validated further by developing customised KA methods for various ESE applications. Besides COBRA (see above) Fraunhofer IESE has multiple projects ongoing in which KA methods for specialised ESE tasks are being developed. One is the elicitation of descriptive software process models [25]. Others are the acquisition of application prerequisites for SE technologies [26] [18] and the validation of software process improvement methodologies [27] [28]. Past experience supports the appropriateness of the methodology. A detailed validation study is currently ongoing (cf. [27]). Future validation results will be reported in the web [29].

4.7 Tool Support for KA in ESE

The application of a KA methodology for ESE raises the question for possible tool support. In general, whether appropriate tooling is readily available depends very much on the specific kind of KA task to be supported (cf. Table 1). For some tasks SE-specific tools are available (e.g., for requirements engineering and process assessment). For other tasks tools from knowledge engineering can be adapted to ESE (cf. [30]). However, for the majority of KA applications in ESE, effective tool support is still rare. This is probably due to the widely lacking formalisation of knowledge management and problem solving processes in ESE. We expect that the KA methodology for ESE presented above will allow for further formalisation and tool support of knowledge acquisition in ESE (cf. *KONTEXT* [18]).

5 Conclusions and Future Work

Many ESE tasks involve some kind of KA activity. These activities can be expected to become more effective, if advanced KA methods will be used in a systematic manner. Therefore KA methods need to be developed that are customised to the specific requirements of the respective ESE tasks. In general, the dissemination of KA methods in ESE should be fostered.

We have provided a survey of KA applications in ESE. It shows that these applications can have very different characteristics and that they impose quite different requirements on the KA method to be used. For this reason, a methodology has been presented for the development of customised KA methods. The design of such methods is facilitated by an experience base of KEL techniques. The experience base represents an ESE-specific knowledge management application. It is accessible for consultants and researchers within Fraunhofer IESE. It supports the dissemination of KEL techniques in experimental software engineering and the accumulation of experiences about their use.

The approach presented in this paper has been and is being applied at Fraunhofer IESE. Main focus of our work is the acquisition of experiential knowledge in experimental software engineering. Example applications are the acquisition

of product/process dependency models [28], lessons learnt about software engineering processes [31], and application prerequisites for SE technologies [26]. Related work addresses the elicitation of software process models [25] as well as cost estimation, benchmarking, and risk assessment of software projects [8].
Future work will address the extension of the experience base of KEL techniques. Additional KEL technologies and experience statements will be added. We have also started to implement the experience base using a knowledge management tool infrastructure. This will allow for case-based knowledge retrieval and interactive decision support for selecting KEL techniques. We are also continuing to develop further our suit of customised KA methods (cf. [24]). Additional information about these activities is provided in [29].

Acknowledgements

We would like to thank Ulrike Becker-Kornstaedt, Frank Bomarius, Lionel Briand, Khaled El Emam, Wolfgang Müller, and Barbara Paech for their valuable feedback on earlier versions of this paper. Sonnhild Namingha has helped us very much to improve our use of the English language.

References

1. H. Dieter Rombach, Victor R. Basili, and Richard W. Selby, editors. *Experimental Software Engineering Issues: A critical assessment and future directions*. Lecture Notes in Computer Science Nr. 706, Springer–Verlag, 1992.
2. 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.
3. Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat, editors. *Building Expert Systems*. Addison-Wesley, 1983.
4. Mark A. Musen. An overview of knowledge acquisition. In David et al. [32], pages 405–427.
5. Robert L. Glass. Expert systems: Failure or success? *Journal of Systems and Software*, 43(1):1–2, October 1998.
6. Peter Grogono. Software engineering for expert systems. In Jay Liebowitz, editor, *The Handbook of Applied Expert Systems*, pages 25–1–25–15. CRC Press, 1998.
7. 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.
8. Lionel C. Briand, Khaled El Emam, and Frank Bomarius. COBRA: A hybrid method for software cost estimation, benchmarking, and risk assessment. In *Proceedings of the Twentieth International Conference on Software Engineering*, pages 390–399, Kyoto, Japan, April 1998. IEEE Computer Society Press.
9. David N. Wilson and Tracy Hall. Perceptions of software quality: A pilot study. *Software Quality Journal*, 7(1):67–75, 1998.
10. J. Angele and R. Studer. Requirements specification and model-based knowledge-engineering. *Softwaretechnik-Trends: Mitteilungen der GI-Fachgruppen 'Software-Engineering' und 'Requirements-Engineering'*, 15(3):4–16, October 1995.

11. Midred L. G. Shaw and Brian R. Gaines. Requirements acquisition. *IEEE Software Engineering Journal*, pages 149–165, May 1996.
12. Klaus Weidenhaupt, Klaus Pohl, Matthias Jarke, and Peter Haumer. Scenarios in system development: Current practice. *IEEE Software*, March/April 1998.
13. Neil A. Maiden and Cornelius Ncube. Acquiring COTS Software Selection Requirements. *IEEE Software*, 15(2):46–56, March 1998.
14. Elizabeth S. Cordingly. Knowledge elicitation techniques for knowledge-based systems. In Dan Diaper, editor, *Knowledge Elicitation: Principles, Techniques and Applications*, chapter 3, pages 89–176. Ellis Horwood, 1989.
15. Nancy J. Cooke. Varieties of knowledge elicitation techniques. *International Journal of Human-Computer Studies*, 41(6):801–849, 1994.
16. M. Welbank. Knowledge acquisition: a survey and british telecom experience. In T. Addis, J. Boose, and B. Gaines, editors, *Proceedings of the First European Workshop on Knowledge Acquisition for Knowledge Based Systems*. Reading University, 1987.
17. R.R. Hoffman, N.R. Shadbolt, M.A. Burton, and G. Klein. Eliciting knowledge from experts: A methodological analysis. *Organizational Behaviour and Human Decision Processes*, 62(2):129–158, 1995.
18. Andreas Birk and Felix Kröschel. A knowledge management lifecycle for experience packages on software engineering technologies. Technical Report IESE-Report No. 007.99/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, 1999.
19. Felix Kröschel. A system for knowledge management of best software engineering practice. Master's thesis, University of Kaiserslautern, Kaiserslautern, Germany, November 1998.
20. Klaus-Dieter Althoff, Frank Bomarius, and Carsten Tautz. Using case-based reasoning technology to build learning organizations. In *Proceedings of the the Workshop on Organizational Memories at the European Conference on Artificial Intelligence '98*, Brighton, England, August 1998.
21. Klaus-Dieter Althoff, Andreas Birk, Christiane Gresse von Wangenheim, and Carsten Tautz. Case-based reasoning for experimental software engineering. In Mario Lenz, Brigitte Bartsch-Spörl, Hans-Dieter Burkhard, and Stefan Wess, editors, *Case-Based Reasoning Technology - From Foundations to Applications*, number 1400, chapter 9, pages 235–254. Springer-Verlag, Berlin, Germany, 1998.
22. Christiane Gresse von Wangenheim, Alexandre Moraes Ramos, Klaus-Dieter Althoff, Ricardo M. Barcia, Rosina Weber, and Alejandro Martins. Case-based reasoning approach to reuse of experiential knowledge in software measurement programs. In Lothar Gierl, editor, *Proceedings of the Sixth German Workshop on Case-Based Reasoning*, Berlin, Germany, 1998.
23. Markus Nick and Carsten Tautz. Practical evaluation of an organizational memory using the goal-question-metric technique. In *Proceedings of the Workshop on Knowledge Management, Organizational Memory and Knowledge Reuse during Expert Systems '99 (XPS-99)*, Würzburg, Germany, March 1999.
24. Andreas Birk. A knowledge acquisition methodology for use in experimental software engineering. Technical Report IESE-Report No. 062.98/E, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern (Germany), 1998.
25. Ulrike Becker, Dirk Hamann, and Martin Verlage. Descriptive Modeling of Software Processes. In *Proceedings of the Third Conference on Software Process Improvement (SPI '97)*, Barçelona, Spain, December 1997.

26. Andreas Birk. Modelling the application domains of software engineering technologies. In *Proceedings of the Twelfth IEEE International Automated Software Engineering Conference*. IEEE Computer Society Press, 1997.
27. 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.
28. PROFES. ESPRIT project 23239 (Product-FOcused improvement of Embedded Software processes). http://www.ele.vtt.fi/profes/.
29. AXIS. Acquisition of Experiential Knowledge in Software Engineering. http://www.iese.fhg.de/axis.html.
30. Barbara Dellen, Frank Maurer, Jürgen Münch, and Martin Verlage. Enriching software process support by knowledge-based techniques. *International Journal of Software Engineering & Knowledge Engineering*, 7(2):185–215, 1997.
31. Andreas Birk and Carsten Tautz. Knowledge management of software engineering lessons learned. In *Proceedings of the Tenth Conference on Software Engineering and Knowledge Engineering*, San Francisco Bay, CA, USA, June 1998. Knowledge Systems Institute, Skokie, Illinois, USA.
32. Jean-Marc David, Jean-Paul Krivine, and Reid Simmons, editors. *Second Generation Expert Systems*. Springer, 1993.