How Experience Management Can Benefit from Relationships among Different Types of Knowledge

Markus Nick, Klaus-Dieter Althoff, Thomas Avieny, Björn Decker

Fraunhofer Institut Experimentelles Software Engineering (IESE)
Sauerwiesen 6, 67661 Kaiserslautern, Germany
nick@iese.fhg.de

Abstract. In a learning organization, knowledge and experience is created and used at different levels of granularity and maturity. However, these different knowledge/experience types usually coexist without any links and relationships. The field of situated cognition shows that such relationships are typical and important in the human learning "procedure" (e.g., in expert-novice learning/teaching).

We propose that experience management systems can benefit from the support of such relationships. The development of such systems includes -in addition- the definition of an appropriate knowledge life-cycle model describing the mentioned relationships (here by the example of best practices and lessons learned), embedding this in business processes, an operative definition of maturity/validity, and respective knowledge representation issues. Such a development results in a concept that can be implemented with commercially available case-based reasoning tools. We illustrate the approach with real-life examples from systems that already exist or are being developed.

1 Introduction

In a learning organization, explicit knowledge is created and used at different levels of granularity. For example, experiences can occur as best practice (more mature, more general, more consolidated) and as lessons learned (more situated).

While "traditional" CBR systems usually have one type of cases at the same level of granularity and maturity (e.g., problem-solution pairs), some more recent systems also manage different types of knowledge and experience at different granularity and maturity levels, but the different knowledge types co-exist without any links (e.g.; SIMATIC: FAQs, manuals, etc. [9]; our in-house experience base COIN [3] in the beginning in 1999: business process descriptions (more mature) and lessons learned (more situated) without links between to the two types of experience).

The challenge for experience management is to go one step further: To use knowledge and experience at a finer granularity to enrich and improve knowledge/experience at more coarse-grained granularity and higher maturity, or to keep up-to-date and/or start knowledge development at this level at all. While the more mature knowledge/experience is changed rather infrequently, the finer grained experience is gained almost daily. This leads to an almost continuous stream of such experience, which has to be handled.

We expect that combining the different types of knowledge and experience offers opportunities for a more continuous learning and improvement of the explicit knowledge and experience in a company. This allows to add more value, e.g., to more coarse-grained best practices by lessons learned (see Fig. 1). Obviously, this principle is particularly beneficial (1) for domains where the more mature, coarse-grained knowledge/experience (e.g., best practice in the form of processes, products, and technologies) is well-documented and accepted and (2) for domains where best practice is still to be developed.

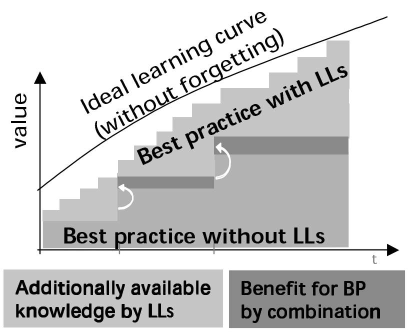

Ideal learning curve (without forgetting)
value
Best practice with LLs
Best practice without LLs
t
Additionally available knowledge by LLs
Benefit for BP by combination

Fig. 1. Development of add-on value of lessons learned (LLs) for a specific best practice (BP) over time.

This idea is related to the field of situated cognition [8]. Situated cognition describes, e.g., the differences in the ways how experts and how novices learn and work [10]: While novices learn and "slavishly" apply best practice (e.g., how to perform a project in a company), experts adapt best practice when they apply it. This adaptation is based on their former experience with the best practice. During the application, they again gain more experience. Thus, a sole description of best practice alone would be "outdated" soon. To avoid this, the best practice descriptions are enriched with more fine-grained experiences gained in the application of the best practice. When re-applying some best practice, the expert is provided with the explicit experiences gained by his colleagues and himself in former applications.

Our in-house experience base COIN [3], which has been running for almost three years, currently contains experiences at three different levels of maturity: business processes (= best practice); general guidelines (situated regarding the business process, but (almost) general regarding projects); and situated lessons learned. This leads to a number of related open issues, which are typical for such systems: When do situated lessons learned become general guidelines? When are lessons learned and/or guidelines integrated into a new/updated business process description? This problem also leads to decision problems in the acquisition of new experience: What should be added as general guideline? What should be added as situated LL? Thus, respective decision criteria are required.

In this paper, we address the issues of how this systematic improvement of knowledge works in principle, how the respective knowledge processes are integrated in business processes, and which are the implications for the knowledge representation and further tool support (as well as CBR as a EM principle).

The experience life-cycle model defines the principle for the development and improvement of the knowledge in the experience base. Here, we focus mainly on the knowledge in the form of cases. The experience life-cycle model allows to identify the relationships of the different knowledge types and the decisions points for the maintenance, i.e., when to transform one type of knowledge into another (Section 2).

This systematic development of knowledge according to the life-cycle model requires that -during the usage of the experience management system- feedback and further experience on the application/usage of the retrieved knowledge is collected. This requires a tight integration of the knowledge-related processes into the business processes. This integration also enforces the systematic validation of experience in different situations/contexts (Section 3).

Having experiences linked with different situations/contexts leads to the need for aggregating the situative information to give the user a better overview on the applicability of the retrieved experience cases (Section 4). In addition, this also reduces the algorithmic complexity of the intelligent search and allows the application of algorithms which are only available for flat case structures.

The paper ends with a summary and conclusion (Section 5).

2 Experience Life-Cycle Models and Validity Issues

A *knowledge/experience life-cycle model* describes the basic idea of how to maintain and improve knowledge and experience over time. Thus, it is the basis of any further refinement of the maintenance/improvement process. A life-cycle model addresses two major issues: (1) the life-cycle of experience of one type (e.g., [11]) and (2) the life-cycle over different types of experience (e.g., [2]). While the life-cycle of cases of one type mainly addresses validity issues and/or its revision status [11], the life-cycle over different types of experience describes when and how experiences are transformed from one type of experience into another (e.g., a number of related lessons learned into a business process model).

2.1 Best Practice and Lessons Learned Life-Cycle Model

We have composed a comprehensive life-cycle model for best practices and lessons learned from aspects presented in [2, 6]. This life-cycle model is applicable to software engineering knowledge and is the basis for many of our experience management projects (e.g., COIN, SKe, InDiGo, ESERNET). In addition, the life-cycle model is sound with the "ideas" of the philosopher Immanuel Kant regarding theory knowledge and experience.

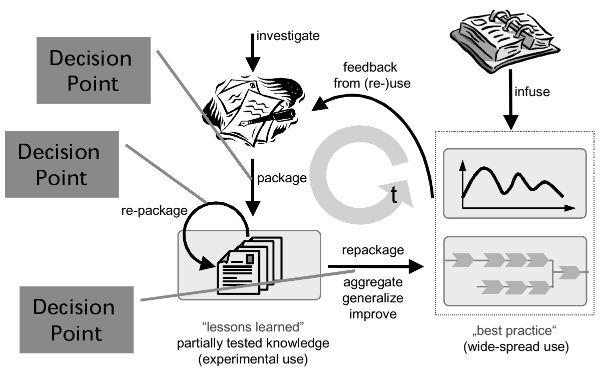


Fig. 2. Generic knowledge life-cycle model for different types of experience.

Fig. 2 depicts this experience life-cycle model for (1) best practice, which is more mature (e.g., business process models and more formal cost models), and (2) lessons learned, which is less mature and more situated experience.

The life-cycle model resembles the way experts apply best practice (see Section 1) and moves this principle from a personal to an organizational level: Existing best practice and the related lessons learned are applied in projects. Based on the feedback from the application in the projects, new experience on the application of the best practice is gained and recorded as lessons learned ("package"). In addition, experience on the application of the existing lessons learned is gained. This experience is used for improving these existing lessons learned ("repackage"). When "appropriate", lessons learned are combined with the best practice description ("aggregate", "generalize", "improve"). E.g., a group of related lessons learned can be transformed into a best practice description that is more comprehensive, mature, and for a broader audience.

Furthermore, new best practice descriptions can be acquired from sources outside the organization (books, conferences, web) - i.e., "infuse". In this case, the organization is similar to a novice. There is no experience available regarding the application of the new best practice in the context of the organization.

There are a number of decision points in the life-cycle model: What should be packaged as lessons learned (including required quality)? When to repackage lessons learned? When to transform or integrate lessons learned into best practice? This also impacts the design and usage of the experience management system (see Section 3). Furthermore, an operational definition of validity and/or maturity is expected to provide better decision support for such life-cycle-related decisions.

2.2 Notions of Validity

There are different notions of validity depending on the audience. We analyse validity from the viewpoint of the users who query an experience base and from the viewpoint of the experience base maintainers.

Validity for users (for retrieval result analysis): The validity describes how much one can trust the experience to be successfully reused and applied in its anticipated new application context. This definition is similar to the definition in [12] where validity is a component of the interestingness of retrieved information from the users' viewpoint. This definition is also related to the definition of external validity in experimentation (i.e., degree to which the results of the research can be generalised to the population under study and other research settings).

Validity for maintainers: The validity provides decision support for maintenance for identifying invalid ("wrong") experience and identifying matured experience(s) that should be transformed into another type of experience.

Operational Definition of Validity: To integrate validity issues into the life-cycle model, operational definitions of validity are required. Such definitions can be in a qualitative or quantitative manner.

An operational definition should consider the following issues. The users' validity should be expressed as one value that is easy to understand regarding its semantics and it should be possible to restrict the search based on validity (e.g., only experiences with a validity ratio > 80% are retrieved). For the maintainers' validity, the operational definition should result in a metric that can be used as input for maintenance decisions support components (e.g., the maintenance assistant from [14]).

To operationalize the above definitions, we use the following quantitative notions of generality and significance, which define certain aspects of validity: *Generality* is the extent of different contexts to which an experience has been applied successfully. *Significance* is the number of successful applications of an experience in a context. Another aspect is *maturity* which we see as the combination of generality and significance.

For already known experiences that are entered into the experience base after an unidentifiable number of applications (e.g., "expert advice" as generalized, aggregated knowledge from an expert's long experience with a certain best practice), a purely quantitative definition of validity obviously cannot be complete because the number of applications before the recording in the experience base cannot be determined. Instead, a qualitative definition of validity is more appropriate. The question is if these qualitative aspects can be quantified to ease analysis.

The operational definitions of validity and the related "measures" obviously require that (re-)use can be measured. The integration of this measurement into the usage is considered in the next section.

3 Integrating Improvement of Knowledge into Usage

To integrate the improvement of knowledge into usage, it is necessary to have both best practice and lessons learned actively used in the "daily business." Here we focus on the lesson learned part.

The idea is to tightly link the usage and evaluation of lessons learned with the application of best practice, which is described by a principle that we call *experience feedback loop*. In each best practice, points can be identified where the application of lessons learned is beneficial, where new experience can be acquired, and where user feedback can be collected, which is used for improving the experience base contents (i.e., "cases").[1]

An experience feedback loop "implements" the life-cycle model and combines this with the measuring of validity as well as with the following ideas: (1) Improvement and maintenance is based on/guided by evaluation through user feedback [14]. (2) The collection of user feedback is "built" into the usage processes to cope with the problem that users are usually not very motivated for entering measurement data during usage [13]. (3) The user is integrated/involved in the systematic improvement of an experience base. (4) The feedback loop can be promoted and enforced by an experience factory as separate organizational unit [5] if the processes are not sufficiently "lived" in the organization.

Fig. 3 shows a general experience feedback loop for lessons learned and illustrates this with examples from project management. At points in time such as the beginning of the application of a best practice (e.g., start of project set-up), the user retrieves the

[1] Feedback can also address the ontology including similarity measures. The reader is referred to [4] for details on the different types of user feedback in experience bases.

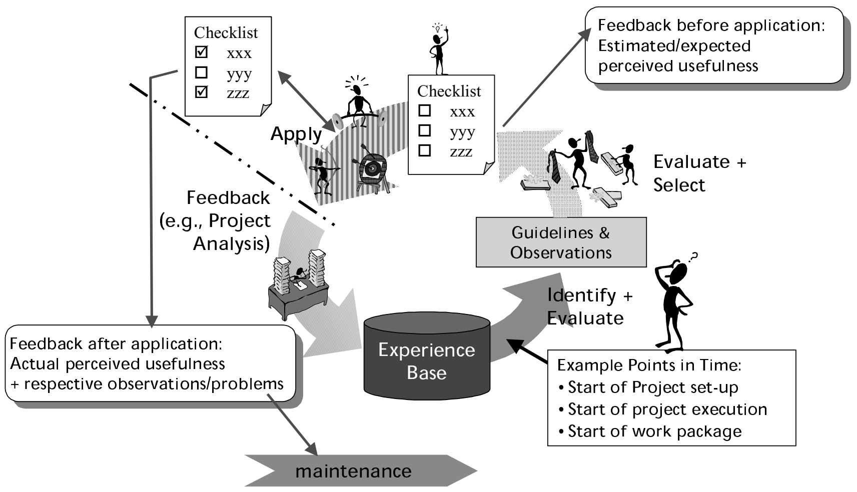


Fig. 3. An experience feedback loop implements the life-cycle model and includes the collection of feedback for improvement/maintenance (examples from the project process in IESE's COIN).

lessons learned that are relevant in his current situation which is mainly characterized by the state of the project (i.e., the point in time) and the project characteristics. He evaluates them considering their applicability, which is described by similarity, validity, significance, and/or generality; and selects the lessons learned he regards as potentially useful. This selection gives an initial feedback on the estimated/expected usefulness of the retrieved lesson learned. The selected lessons learned are used as a kind of checklist during the application of the best practice. After the application of the best practice, feedback is collected to (a) further validate the lessons learned, (b) identify problems with the best practice (which are stored as lessons learned), and (c) record new lessons learned (in general) - considering predefined quality criteria for lessons learned. This feedback is the basis for further maintenance operations on the experience base.

Such a feedback loop has been installed for our in-house experience base COIN [7] where we collect lessons learned in the form of guidelines, observations, problem-solution pairs, etc. that are related to the project process (as collection of best practices). Currently, we tailor the general feedback loop for IT security experience bases in the publicly funded project SKe.

The feedback loop has consequences for the experience representation: Since lessons learned are more or less successfully applied in different situations/contexts, lessons learned will have more than one application context to which they are related.

4 Representation with more than one Context

For software engineering experiences, we have already developed a representation that allows to have lessons learned with more than one application context [1]. Typically, the application context is defined through the related projects in which a lesson learned was applied. In addition, the lessons learned are also related to the respective business processes or software processes (Fig. 4).

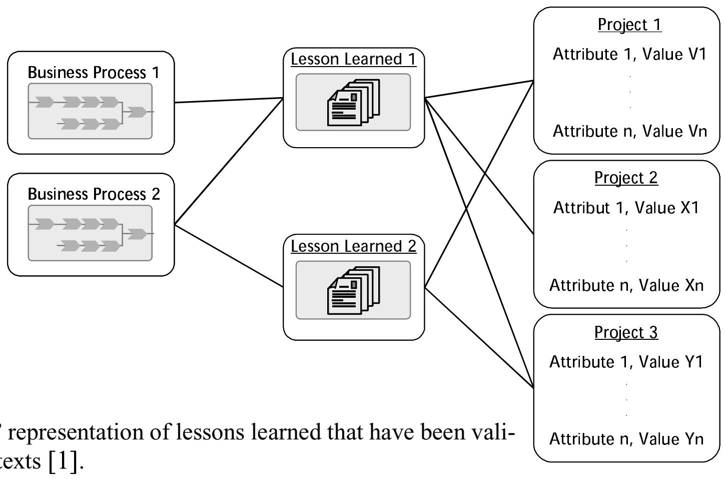


Fig. 4. "Traditional" representation of lessons learned that have been validated in several contexts [1].

The lessons learned can be linked to a number of projects and each project can be linked with a number of lessons learned. To avoid inconsistencies, projects and lessons learned are stored in different cases and references are used to establish the links among them.

The feedback loop has the following consequences: Each time, an experience is successfully applied in another situation/context, the experience is linked with a description of this new situation/context (e.g., project description). Thus, more and more situations/contexts are linked with the experiences. In the following, we will show that this leads to comprehensibility problems when a user wants to analyse the applicability, validity, etc. As a solution, we present a principle and techniques for context aggregation.

4.1 The Comprehensibility Problem

For the user who analyses a retrieval result, the comprehensibility of a lesson learned decreases with an increasing number of a projects linked with the lesson learned. This is because a search result shows either the lesson learned and the most similar project[2] or the lesson learned and all related projects (e.g., Projects 1 and 3 for Lesson Learned 2 in Fig. 4). In both cases, it is difficult for the user to find out how much he can trust the lesson learned in his current situation and how general the lesson learned is – for example: For the degree of validity in his current situation, he has to check each of the projects and find out if it is more or less similar. For the generality, he has to analyse how diverse the projects are.

4.2 Context Aggregation

To improve comprehensibility, the contexts (i.e., projects) that are linked with a lesson learned are aggregated to one single context that is stored with the lesson learned itself

[2] This cannot be implemented with CBR-Works or Orenge.

(Fig. 5). The difficulty is to find an appropriate representation of the aggregated attribute that is easy to use and understand and has a meaningful semantics.

Because the aggregation requires additional attributes and code for performing the aggregation, this will usually only be done for selected attributes.

The aggregation has also an additional benefit regarding the system performance if the aggregation is "static", i.e., does not depend on the query: Assuming that access to cases (e.g., in a database) is the "bottleneck", the complexity of the search over lessons learned is now linear instead of quadratic/polynomial. Using aggregation, only the lessons learned have to be accessed in a search over lessons learned (linear complexity). Without aggregation, lessons learned and related projects have to be accessed (quadratic complexity).

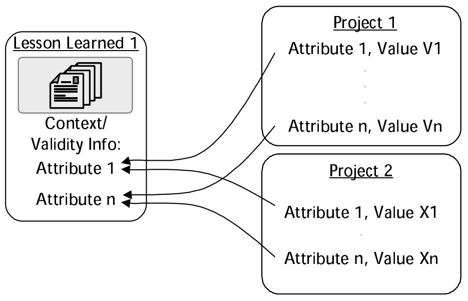


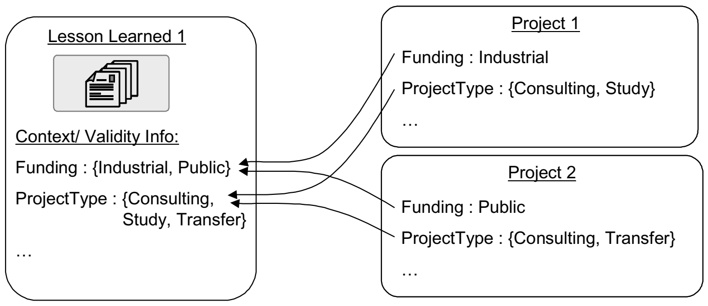
Fig. 5. Aggregation of context of lessons learned.

4.3 Exemplary Context Aggregation Techniques

In the following, two different aggregation techniques and their characteristics are presented and discussed. This includes the kind of attribute types that are supported and for CBR systems: How a query could be specified using the aggregated context and which similarity measures are required. The techniques assume that an experience item is only linked with situations where the application was successful. If the aggregated attribute would include unsuccessful applications, the attribute would not reflect the trust issue from above.

4.3.1 Aggregation using Union Function

Attribute types that are discrete and have a limited value range can be aggregated using the union function for sets. The aggregated value is the union of the values from the attributes in all related situations (Duplicates are eliminated). Fig. 6 shows this by an example for lessons learned and projects.



Lesson Learned 1
Context/ Validity Info:
Funding : {Industrial, Public}
ProjectType : {Consulting, Study, Transfer}
…
Project 1
Funding : Industrial
ProjectType : {Consulting, Study}
…
Project 2
Funding : Public
ProjectType : {Consulting, Transfer}
…

Fig. 6. Aggregation using a set union function by the example of lessons learned and projects.

Supported Attribute Types: Symbolic types, symbol sets, numerical types.

Numerical types require some preprocessing for an efficient handling: Intervals from the type value range are mapped to symbols (e.g., mapping 1..5 on “1-5”, 6..10 to “6-10”, 11..∞ to “>10”). These symbols can then be aggregated as described above. By mapping N..∞ or −∞..N to a symbol, even numeric types with unlimited value ranges can be aggregated. Using such a mapping, non-discrete numeric types can also be aggregated.

Specification of Context in Query: The “wanted” values of the attributes in the aggregated context are specified as set (i.e., the same as for any other attribute).

Similarity in CBR systems: Standard similarities for sets can be applied.

Characteristics:

- The union function approach models the generality of an experience, i.e., to which different contexts it has been applied.
- A distinct validity cannot be expressed because the aggregated attribute value shows only if the experience was applied in a context where the attribute had one or several of the values from the aggregated value. That is, the aggregated validity would be binary (yes/no). Thus, expert advice cannot be modelled meaningfully.
- Representation and algorithm are simple. The algorithm is obviously “static”, i.e., does not depend on a query specification.

The union-function aggregation is suitable for symbol types where the audience focuses on the aspect of generality rather than significance. The type range can be of any size. However, the larger the range, the more difficult it becomes to have a really general experience.

The described aggregation will be used in the representation of a repository on software engineering technologies and lessons learned in the European project ESERNET.

4.3.2 Significance-Based Aggregation

The significance-based aggregation includes the aspect of validity. The significance is defined at the value level, i.e., each value of a context attribute as a different significance. For this purpose, we define the absolute and relative significance as follows:

$$significance_{absolute}(\text{LL,attr,value}) = \text{\# of links to situations where LL.attr = value}$$

To normalize the significance in a meaningful way so that the value can be used in, e.g., similarity calculations, we define a relative significance that relates the absolute significance to a maximal absolute significance. Like the validity, the maximal absolute significance depends on the viewpoint, i.e., maintainer or retrieval user. This is included as a parameter “X” in the following definition. For retrieval, “X” is the set of the values of the attributes used in the query specification (i.e., the significance of an LL is compared to the significance of lessons learned in the same or similar context as defined by the query). For maintenance, “X” is all values of all attributes used in aggregation.

$$significance_{relative}(\text{LL,attr,value}) = \frac{significance_{absolute}(\text{LL,attr,value})}{max_X[(\forall \text{LL})significance_{absolute}(\text{LL,attr,value})]}$$

Supported Attribute Types: Symbolic types, numeric types (same approach as for union function aggregation).

Specification of Context in Query: Assuming a normalization to [0..1], a significance value of 1.0 is specified for each "wanted" value of the attributes in the aggregated context. Any other value is excluded from the search (i.e., specified as "undefined" in CBR-Works/orenge).

Similarity in CBR systems: The normalized relative significance suggests to used the respective standard similarity measure, i.e., sim(q,i) := (q-i) / 1.0. Only significance attributes for "wanted" values are considered for the overall similarity of the lesson learned.

Characteristics:

- The significance-based aggregation models generality and validity. An "estimate" validity ratio is calculated with the similarity of the aggregated context. This validity refers to the successful applications in contexts that are similar to the query context.
- A low number of different values in the value range of the aggregated type is desirable because a respective significance attribute must be added to the aggregated context for each value of an attribute from the context. For a higher number of different values, such a context description is obviously hard to comprehend and, therefore, not adequate.
- Expert advice can be modelled by linking a lesson learned with a context description that contains estimated absolute significance values according to the estimated significance of the expert advice.
- The algorithm is "static" (i.e., does not depend on a query specification). For this purpose, the maximum values of each value of each attribute in the aggregated context have to be stored. Then, in CBR systems, the significance can be expressed as a similarity measure as described above.

The significance-based aggregation is particularly suitable for symbol types with a low number of symbols or other types that can be mapped on a low number of discrete values.

5 Conclusion

We have presented an approach that allows to systematically link different types of experience at different levels of granularity and maturity. These different types can complement each other and are used for the systematic maintenance and improvement of the experiences. This approach was demonstrated by examples with two degrees of granularity - best practice and lesson learned.

The approach is holistic, i.e., based on a cognitive principle an experience life-cycle model describes the systematic improvement of experience, which is integrated into daily work using an experience feedback loop. An operational definition of validity and context aggregation address the resulting representational and implementation issues.

Thus, the approach moves a cognitive principle from a personal to an organizational level.

The approach is "lean", which means that it can be implemented with existing tools with little additional implementation development and without changes to the existing tools (e.g., using the commercially available CBR tools such as cbr:works or orenge from empolis/tec:inno GmbH). The feedback loop requires the integration of feedback data collection into the system. In [14], we have presented the respective representation and architecture. The aggregation requires an additional tool/component that performs the respective computation whenever a case or link is added, modified, or removed. Furthermore, the described aggregation reduces the computational complexity of a typical similarity-based search algorithm from CBR and allows the application of algorithms which are only available for flat case structures.

Specific elements of the approach are used in various projects at our institute (e.g., the in-house experience base COIN; the public projects SKe, InDiGo, and ESERNET). An extension of the approach by adding an even finer grained discussion level in addition to lessons learned and best practice is developed in the public project InDiGo.

Acknowledgements

The COIN project has been funded as an internal project at IESE since January 2000. The SKe project (holistic security concepts with dynamic control mechanisms for eService processes) is funded by German government under contract No. 01AK900A running 2001-2003. The InDiGo project (integrative software engineering using discourse-supporting groupware - http://ais.gmd.de/~leopold/indigo.html) is funded by German government under contract No. 01AK915A. running 2001-2003. The ESERNET project (Experimental Software Engineering Network - http://www.esernet.org/) is funded by the European Union (Project IST-2000-28754).

References

[1] K.-D. Althoff, A. Birk, S. Hartkopf, W. Müller, M. Nick, D. Surmann, and C. Tautz. Systematic population, utilization, and maintenance of a repository for comprehensive reuse. In G. Ruhe and F. Bomarius, editors, *Learning Software Organizations - Methodology and Applications*, number 1756 in Lecture Notes in Computer Science, pages 25–50. Springer Verlag, Heidelberg, Germany, 2000.

[2] K.-D. Althoff, F. Bomarius, and C. 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, Aug. 1998.

[3] K.-D. Althoff, B. Decker, S. Hartkopf, A. Jedlitschka, M. Nick, and J. Rech. Experience management: The fraunhofer iese experience factory. In P. Perner, editor, *Proceedings of the Industrial Conference Data Mining*, Leipzig, Germany, 2001. Institut für Bildverarbeitung und angewandte Informatik.

[4] K.-D. Althoff, M. Nick, and C. Tautz. Improving organizational memories through user feedback. In *Workshop on Learning Software Organisations at SEKE'99*, Kaiserslautern, Germany, June 1999.

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

[6] V. R. Basili and H. D. Rombach. Support for comprehensive reuse. *IEEE Software Engineering Journal*, 6(5):303–316, Sept. 1991.

[7] M. Brandt, D. Ehrenberg, K.-D. Althoff, and M. Nick. Ein fallbasierter Ansatz für die computergestützte Nutzung von Erfahrungswissen bei der Projektarbeit. In H. U. Buhl, A. Huther, and B. Reitwiesner, editors, *Information Age Economy. 5. Internationale Tagung Wirtschaftsinformatik 2001*, pages 251–264, Heidelberg, Germany, 2001. Physica-Verlag.

[8] W. J. Clancey. *Situated Cognition — On Human Knowledge and Computer Representations*. Cambridge University Press, Cambridge, UK, 1997.

[9] M. Lenz, K.-H. Busch, A. Hübner, and S. Wess. The simatic knowledge manager. In D. W. Aha, I. Becerra-Fernandez, F. Maurer, and H. Munoz-Avila, editors, *Exploring Synergies of Knowledge Management and Case-Based Reasoning (Technical Report WS-99-10)*, pages 40–45. AAAI Press, 1999.

[10] T. Menzies. Knowledge maintenance: The state of the art. *The Knowledge Engineering Review*, 14(1):1–46, 1998.

[11] M. Minor and A. Hanft. Corporate knowledge editing with a life cycle model. In *Proceedings of the Eighth German Workshop on Case-Based Reasoning*, Laemmerbuckel, Germany, 2000.

[12] M. Müller. Interestingness during the discovery of knowledge in databases (in German). *Künstliche Intelligenz*, pages 40–42, Sept. 1999.

[13] M. Nick and K.-D. Althoff. Engineering experience base maintenance knowledge. Technical Report IESE-Report No. 018.01/E, Fraunhofer IESE, 67661 Kaiserslautern, Germany, 2001.

[14] M. Nick, K.-D. Althoff, and C. Tautz. Systematic maintenance of corporate experience repositories. *Computational Intelligence - Special Issue on Maintaining CBR Systems*, 17(2):364–386, May 2001.