Improving Knowledge Acquisition in Capstone Projects Using Learning Spaces for Experiential Learning

Eric Ras
Fraunhofer IESE
Fraunhofer-Platz 1
67663 Kaiserslautern, Germany
eric.ras@iese.fraunhofer.de

Joerg Rech
Fraunhofer IESE
Fraunhofer-Platz 1
67663 Kaiserslautern, Germany
joerg.rech@iese.fraunhofer.de

Abstract

Students have to cope with new technologies, changing environments, and conflicting changes in capstone projects. They often lack practical experience, which might lead to failing to achieve a project's learning goals. Reusing experiences from other students provides a first step towards building up practical knowledge and implementing experiential learning in higher education. In order to further improve knowledge acquisition during experience reuse, we present an approach that generates so-called learning spaces, which automatically enrich experiences with additional learning content and contextual information. To evaluate our approach, we conducted a controlled experiment that showed a statistically significant improvement for knowledge acquisition by 204% compared to conventional experience descriptions. From a technical perspective, the approach provides a good basis for future applications that support learning at the workplace in academia and industry.

1. Introduction

Today, professional software engineers have to cope with the continuous change of technologies and their knowledge about these. This change, the short innovation cycles, and the fact that software engineering is a knowledge-intensive activity lead to many learning situations where new knowledge is required to solve the challenges and problems at hand. Furthermore, learning in practice is less a reaction to 'being learned' but more the reaction to a variety of working situations and related problem-solving activities. Most of our daily learning is, in fact, experience-based, and professional working relies on engineers' experience.

From an educational point of view, gathering practical experience in software engineering is the crucial issue of each curriculum. The Software Engineering Curriculum Guidelines (SE2004) [1] also mandate that students undertake a capstone project to expose them to the application domain and practice what they have learned, and that software engineering should be taught as a problem-solving discipline. Practicing SE should be performed on a reflective basis, since the students must learn to judge the application of specific methods and techniques, and to critically evaluate the consequences of their actions and decisions. One way of supporting reflective practice is to use experiential learning as the underlying process [2].

However, students have to cope with several challenges in capstone projects, which often lead to failing to achieve the learning objectives: Many capstone projects risk overloading students because they get overwhelmed with so many new topics, because they have to understand the different roles and responsibilities assigned, and because they have to cope with a changing environment (e.g., software requirements). Teachers want to provide realistic projects and conflicting situations as they happen in the real world to prepare students for their jobs [3]. We have experienced very similar problems in the past during capstone projects at the University of Kaiserslautern.

In order to address the lack of practical experience, we started to let students gather their observations and experiences during the capstone project in an adapted Wiki-based system

called Software Organization Platform (SOP). SOP has also been used for project management and requirements engineering as well as for the coordination of development tasks [4]. Experiences are consolidated in this Experience-Factory-inspired platform by the teachers and fed back into future capstone projects in the following year. However, lack of background knowledge and very brief experience descriptions make it difficult to reuse the experiences in new situations. Therefore, learning at the workplace is supported by a context-aware fusion of previously developed learning content with collaboratively developed documents and experience descriptions from daily work. These so-called learning spaces are intended to enhance experience reuse by following a specific learning goal, and are created based on context information of the experience package.

The work addresses the research question of whether learning spaces could improve the understanding and application of an experience package on the one hand and knowledge acquisition on the other hand. In this paper, the investigated hypotheses are related to the claim that learning spaces lead to higher knowledge acquisition compared to conventional experience descriptions. The approach provides a learning approach for capstone projects in higher software engineering education that relies on experiential learning, as well as an innovative solution for supporting experience reuse during workplace learning in industrial settings.

This paper focuses on the presentation of the design and results of a controlled experiment, whose aim was to investigate the impact of learning space on knowledge acquisition. Section 2 will list previous work and related work about experiential learning. Section 3 briefly presents the approach for generating learning spaces. Section 4 explains the hypotheses, the design of the experiment, the analysis, and the results. Section 5 concludes the paper.

2. Background

The learning space approach uses SOP as base platform for generating and presenting learning resources. Previous studies showed that Wikis are a good medium for content creation and sharing in capstone projects and are very well accepted by students [4]. In addition to the easy documentation of project tasks, observation, and experiences made, the platform has shown its suitability for involving stakeholders in requirements engineering [5].

The underlying learning theory behind our approach is experiential learning, as it is an effective way to construct new knowledge, with students continually going through a learning cycle: “practicing, reflecting on the difficulties, discovering new models (or having them introduced by facilitators or other students), and then practicing again” [6]. Research on experiential learning is based on the work of Kolb and Fry [2, 7], who investigated the learning process related to learning from experiences and whose research has its foundation in the work of Lewin, Dewey, and Piaget. Reflection is the prerequisite for learning from experience (e.g., in order to form abstract concepts) and for improving actions and professional practice [2]. The value of reflection has already been proven in situated cognition theory (e.g., *Cognitive Apprenticeship* and *Anchored Instruction)*.

Capstone projects have shown to support experiential learning [8], where students reflect and interpret their experiences to build abstractions, which are applied and tested in new situations and provide the foundation for new experiences. However, different problems occur when documented experiences are reused by others. Experience is often documented by domain experts. Expert knowledge is somehow ‘routine’ [9]. This makes it challenging for experts to document experiences appropriately and make them reusable for others. Novices lack software engineering background knowledge and are not able to connect the experience to their knowledge base. Hence, they often misinterpret or may not even understand other people’s documented experience. A more detailed summary of problems related to understanding and learning from documented experience can be found, for example, in [10] and [11].

3. Learning space generation approach

To address these problems, an approach has been developed to produce so-called learning spaces for enhancing first, the reuse of experience packages and second, knowledge acquisition. A learning space is generated by the system when a user accesses an experience package (i.e., experience description) from the database during the project. The generation process enriches the experience package with additional situational and instructional content. From a technical point of view, a *learning space* consists of a hypertext document with linked pages. A learning space follows a specific global learning goal (the learning goal level is selected by the student) and is created based on context information about the current situation and the experience package. The learning space is presented by means of Wiki pages within SOP (see Fig. 1). Next, the basic concepts will be briefly explained; however, we refer to [11] for a detailed description of the concepts and the generation process in particular.

The *composition of a learning space* uses an instructional design template with fine-grained learning objectives, which implement the learning process of experiential learning. The approach uses Anderson and Krathwohl's taxonomy of educational objectives [12], which is a revision of the original taxonomy by Bloom. This approach addresses all the cognitive processes, with the focus being on the first three categories (remember, understand, apply), because these are important for reaching the upper levels and can be taught directly, while the fourth to sixth levels (analyze, evaluate, create) require more time and a deeper understanding of a subject matter.

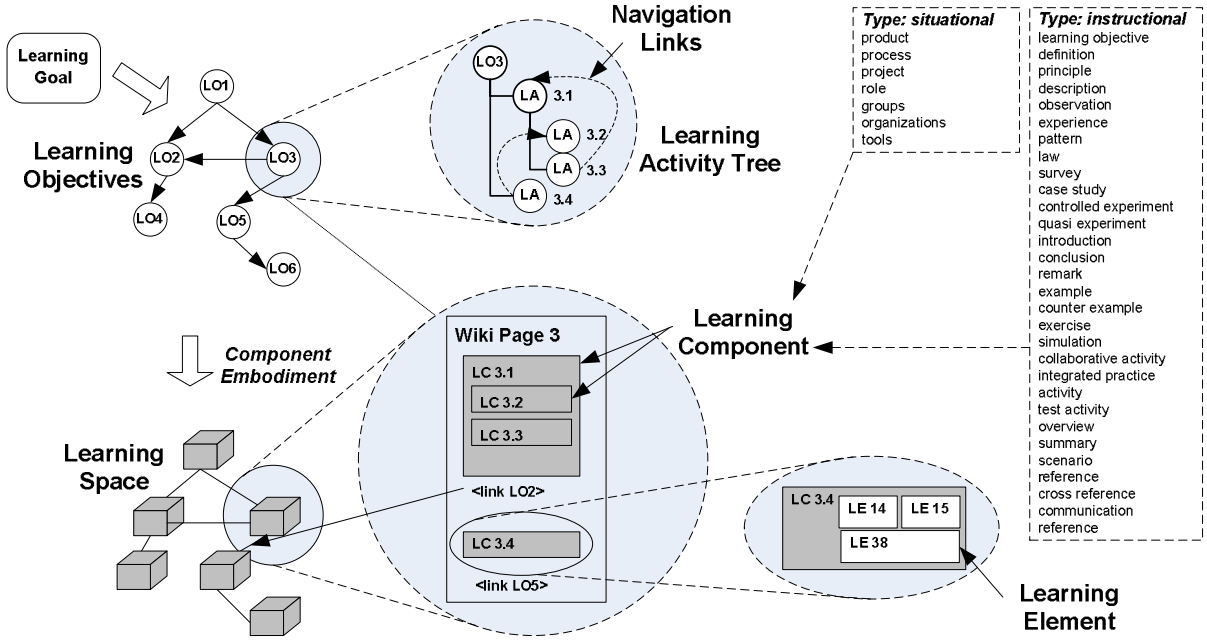

Figure 1. Concepts of a learning space

Before the template is filled with content, each learning objective is refined in a learning activity tree by means of instructional design templates, which are available for each learning objective/concept type pair (e.g., remember/project, understand/product, apply/process, etc.). Each activity tree consists of learning activities that enable the learner to reach the related learning objective (e.g., reading, thinking about a question posed, removing a real code defect, remembering a project, asking a colleague). The embodiment creates learning components from learning elements by using a software engineering domain ontology. *Learning elements* are the most basic learning resources. They are electronic representations of media, such as images,

text, sound, or any other piece of data that can serve as a learning resource when aggregated with other learning elements to form a learning component. *Learning components* are units of instruction that contain at least one learning element. The difference between a learning component and a learning element is that a learning component has a type, either *situational* or *instructional*, and is related to a learning objective, respectively to a learning activity. In addition, it can be referenced as a learning resource by the system (e.g., by using hyperlinks). Situational learning components contain information about the context description of the experience package. They should primarily support the understanding of the experience package. Instructional components are more dedicated to learning experience package related topics and competence development in general.

4. Empirical evaluation

In order to evaluate the claim that learning spaces lead to higher knowledge acquisition compared to conventional experience descriptions, a counter-balanced, one-factorial, within-subject experiment was conducted with 19 undergraduate and graduate students of the University of Kaiserslautern, who took part in a capstone project. The independent variable (factor) was the set of information provided, i.e., experience package (EP) or experience package enriched with learning space (LSEP). The first group (ten subjects) was assigned to use LSEP, while the second group (nine subjects) used EP during the first day. The assignment of the groups was reversed for the second day. In order to prevent undesired sources of variation from being introduced to the experiment, randomization was done: First, regarding the selection and sequence of the experience packages used, two prepared experience packages were selected for each day from a total of eight (i.e., four were not used for the experiment). Second, the subjects were grouped into four levels of experience, based on the results of the briefing questionnaire. The students with the same level of experience were randomly assigned to the experimental and control group.

4.1. Experiment planning and execution

The results presented in this paper rely on the variable knowledge acquisition and not on the specific performance during reuse of the experience (i.e., application efficiency, completeness, and accuracy) because the focus in this paper is about learning from experience packages:

- "Average overall knowledge acquisition difference (kad)" – The average overall knowledge acquisition of the experimental group (LSEP) is higher than the average knowledge acquisition of the control group (EP):
  $H_{1.1}$: $\mu(kad_{LSEP}) > \mu(kad_{EP})$; $H_{0.1}$: $\mu(kad_{LSEP}) \leq \mu(kad_{EP})$
- "Average overall knowledge acquisition difference related to the cognitive knowledge dimension (kad_cog_x)" – The average knowledge acquisition of the experimental group (LSEP) is higher than the average knowledge acquisition of the control group (EP) for the cognitive process dimensions (x) *remembering*, *understanding*, *applying*, *analyzing*, and *creating*:
  $H_{1.2.x}$: $\mu(kad\_cog\_x_{LSEP}) > \mu(kad\_cog\_x_{EP})$; $H_{0.2.x}$: $\mu(kad\_cog\_x_{LSEP}) \leq \mu(kad\_cog\_x_{EP})$

For evaluating the hypotheses, the significance level α was set to 0.05 (error type I) and, the power was assumed to be higher than 0.80.

The learning goals and the assessment tests (i.e., pre- and post-test) of the learning space were classified according to the cognitive process dimension of Anderson and Krathwohl [12]:

Remembering is to promote the retention of the presented material, i.e., the learner is able to retrieve relevant knowledge from long-term memory. The associated cognitive processes are recognizing and recalling. *Understanding* is the first level of promoting transfer, i.e., the

learner is able to construct meaning from instructional messages. He builds a connection between the "new" knowledge to be gained and his prior knowledge. The associated processes are interpreting, exemplifying, classifying, summarizing, inferring, comparing, and explaining. *Applying* also promotes transfer and means carrying out or using a procedure in a given situation to perform exercises or solve problems. The associated processes are executing and implementing. *Analyzing* also promotes transfer and means breaking material into its constituent parts and determining how the parts are related to one another as well as to an overall structure or purpose. The associated processes are differentiating, organizing, and attributing. *Creating* also promotes transfer and is putting elements together to form a coherent whole or to make a product. Learners are involved in making a new product by mentally reorganizing some elements or parts into a pattern or structure not clearly presented before. The associated processes are *generating*, *planning*, and *producing*. It was not possible to cover the level of *evaluating* strongly enough in the learning space. Therefore, this category was omitted.

The design of the experiment is shown in Fig. 2. It was conducted on three subsequent days, with an introduction to the experiment on the first day. Each experience package/learning space was related to the refactoring activity and one specific code smell. Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code, yet improves its internal structure [13]. The results of a briefing questionnaire showed that only four students had average experience in refactoring. During each experimental unit, each subject had to fill out a pre-test of 93 questions [14] (about 50% were yes/no questions or multiple choice questions, which ensured that they could finish on time). Each of these questions was related to one of the different cognitive processes. Afterwards, both groups received the same two experience packages, either conventional or enriched, and were asked to solve assignments based on the information provided. The assignments consisted of practical exercises where they refactored their own code from the project. Appropriate code fragments had been chosen by the evaluators beforehand. Then, the same questions had to be filled out again as post-test. A debriefing questionnaire measured the perceived task complexity, usefulness, and acceptance of the experiment (i.e., disturbing factors). The knowledge acquisition difference was measured by subtracting the pre-test score from the post-test score.

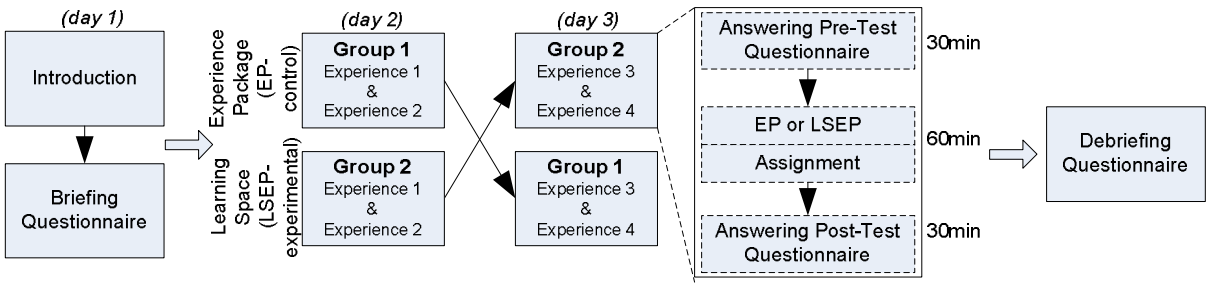

Figure 2. Procedure of the experiment

4.2. Data preparation

An item analysis based on the post-test data of the third day was performed to select the best items for inclusion and to identify poorly written test items. The items were selected based on their discrimination index (D), discrimination coefficient (r), and item difficulty (p). The discrimination index (D) is a measure of a question's ability to differentiate between high and low achievers (i.e., the subjects were assigned to three groups according to their post-test scores of the second day: 27% lower; 46% middle; 27% upper). The index is the number of people in the upper group who answered the item correctly minus the number of people in the lower group who answered the item correctly, divided by the number of people in the largest

group. The discrimination index is based on biserial correlation, which measures whether the scale measured by the test score (i.e., knowledge acquisition) is also measured by the single item. Item difficulty was measured by the percentage of students answering a question correctly. In summary, 28 items were deleted from the test due to negative discrimination indices, very high or low difficulty indices (i.e., resulting in a dispersion that was too low), and low or even negative discrimination coefficients. By doing this, the reliability of the test with 65 items (i.e., 23 remember items, 28 understand items, 10 analyze items, 4 create items) increased from a Cronbach's alpha of 0.665 to 0.812. This reliability can be stated as high because the test captures quite heterogeneous topics and is related to very different cognitive processes. The score of the pre- and post-tests were calculated by summing up the scores of the single items for each cognitive level. The item scores were defined before the experiment by testing the test with other students not involved in the experiment. In order to get the total test score, the cognitive level scores were summed up. Missing values were replaced by zero. Outliers have been removed. The knowledge acquisition difference was calculated by subtracting the pre-test score from the post-test score. The score for the cognitive level *apply* was not assessed by the test but was calculated by using the score that students got for the solution of the assignments, i.e., the score for identifying and removing the code smells.

4.3. Analysis and results

Randomization and counter-balancing are thought to be a solution for avoiding period-, sequence-, and carry-over effects in cross-over designs, as applied in this experiment. However, they are seldom investigated. Sequence effects as well as carry-over effects were detected. Nevertheless, they were not significant statistically. A significant period effect was detected for kad_cog_apply – a correction of the p-value was necessary. The dependent samples t-test requires period differences to possess a normal distribution for both groups. The Shapiro-Wilk test for normality distribution showed that no normal distribution was detected for *kad_cog_understand* and *kad_cog_create* (see Table 1). For these two variables, a non-parametric Wilcoxon matched paired test was performed.

Table 1. Descriptive results of the dependent variables

knowledge acquisition difference	*group*	*N*	*kad mean*	*std. dev.*	*std. error mean*	*Shapiro-Wilk Period Diff (Sig.)*	*Relative improvement*
kad ($H_{1.1}$)	experimental	19	17.95	6.14	1.40	.446	204.75%
	control	19	5.89	4.45	1.02		
kad_cog_remember ($H_{1.2.1}$)	experimental	19	5.63	3.202	.800	.823	211.05%
	control	16	1.81	.981	.245		
kad_cog_understand ($H_{1.2.2}$)	experimental	19	6.11	3.19	0.73	**.043**	274.84%
	control	19	1.63	2.45	0.56		
kad_cog_apply ($H_{1.2.3}$)	experimental	18	12.40	5.328	1.256	.718	46.91%
	control	18	8.44	4.026	.949		
kad_cog_analyze ($H_{1.2.4}$)	experimental	19	4.68	3.31	0.76	.396	128.29%
	control	19	2.05	2.48	0.57		
kad_cog_create ($H_{1.2.5}$)	experimental	18	1.22	1.003	.236	**.016**	117.85%
	control	18	.56	.784	.185		

Looking at the means and the relative improvement rates, the subjects of the experiment group performed better than the control group regarding all variables. The highest improvement is related to the cognitive process *understand*, which was deemed to be crucial for reuse artifacts [15]. In order to find out whether these differences between means are not due to chance, statistical tests were applied (see Table 2 for the results). The effect size γ is a way to describe the strength of a relationship between two variables. In the context of a t-test, the effect size is calculated as the difference between the means of both groups divided by the

pooled standard deviation for those means. A post-hoc power analysis was done based on effect size, sample sizes, and α.

Table 2. Results of the hypothesis tests

	One-tailed dependent samples t-test						*Wilc. test*				
					95% Confidence Interval of the Difference						
	df	*t*	*Crit. $T_{0.95}$*	*p-value*	*lower*	*upper*	*p-value*	*effect size (γ)*	*power (1-β)*	*reject H_0*	*accept H_1*
kad	18	7.168	1.734	.000	8.520	15.585	-	1.64	1.000	yes	yes
kad_cog_remember	15	4.111	1.753	.001	1.836	5.789	-	1.02	0.988	yes	yes
kad_cog_understand	18	4.649	1.734	.000	2.452	6.496	.001	1.06	0.997	yes	yes
kad_cog_apply	17	2.423	1.740	.006*	.512	7.405	-	1.54	0.933	yes	yes
kad_cog_analyze	18	3.324	1.734	.004	.968	4.295	-	0.76	0.939	yes	yes
kad_cog_create	17	2.380	1.740	.029	.076	1.258	.026	0.56	0.739	yes	no

* adjusting for a period effect reduces the p-value from .027 to 0.006 by using an independent samples t-test based on period differences

4.4. Discussion

For all dependent variables, p-values were lower than $\alpha = 0.05$. Hence, all null hypotheses can be rejected. For the total knowledge difference, and the knowledge difference on the remember, understand, apply, and analyze levels, the power is higher than 0.80, which means that we can accept the alternative hypotheses. However, the alternative hypothesis $H_{1.2.5}$ cannot be accepted due to the fact that the power is 0.739. Nevertheless, $H_{1.2.5}$ could probably have been accepted with a slightly larger sample size (we would have needed 22 subjects for $H_{1.2.5}$). In general, learning spaces are better for knowledge acquisition than conventional experience descriptions. The effect sizes passed the level of 0.80 for almost all variables. This means that learning spaces have a "large" effect [16] for at least the lower cognitive levels.

Regarding *construct validity*, measuring knowledge acquisition is difficult. Nevertheless, the test items of the pre-test, post-test, and assignments were related to different cognitive learning goals, and we assessed factual and conceptual as well as procedural knowledge [12]. The item analysis ensures that only items with a strong contribution to the overall knowledge acquisition measuring construct were used for analysis (see Section 4.2). Higher *internal validity* was achieved by counterbalancing and randomization. Furthermore, the correction for the period effect provides a more reliable result for *kad_cog_apply*. Instrumentation effects could be ignored because the assessment of the experiment material's difficulty did not show any differences. One problem of *external validity* is that students are unlikely to be representative of software professionals. The students who participated were undergraduate and bachelor students with an average study time in computer science of 3.5 years and an average software development experience of 4.2 years. All had attended software engineering courses with practical courses lasting three terms and some of them had attended specific SE courses (e.g., quality assurance, process modeling, or project management). Nevertheless, the results can be useful for industrial contexts because software development is often done by people who have just finished their studies and therefore are comparable to the students in this experiment. In addition, refactoring is not a well-known and widely applied technique in industry and the experience descriptions were based on literature intended for practical use in real projects. This means that similar results could be expected in an industrial setting and the results of this experiment can be used as a baseline for further studies in the field. Nevertheless, further evaluation related to other topics is necessary to confirm that the effect does not depend on the topic taught.

5. Conclusion

Software engineers in academia and industry are often confronted with new situations they have never been in. They have to learn in a self-directed way and often by reusing documented experiences from colleagues collected in special knowledge bases such as forums on the Internet or internal Experience Factories.

Our approach provides an innovative learning technique for capstone projects in higher education as well as a solution for supporting workplace learning during experience reuse in industrial settings. The results of our controlled experiment confirm that students acquire 204% more knowledge when using learning spaces than with conventional experience descriptions. The approach is promising for capstone projects, because we can cope with the lack of the students' practical experience by bringing documented experiences, enriched with additional learning content, from more experienced students or practitioners into the learning environment. Furthermore, as the Net Generation is creating, sharing, and searching for more and more information in networks, the Wiki-based technology used in SOP promises a lightweight solution to collaboratively capture, organize, and distribute knowledge that emerges, and to serve as a basis for generating and presenting learning spaces. A technical advantage of this approach is that the generation of learning spaces does not rely on a closed corpus of content, and, hence, can be extended with experience and learning content from other repositories on the WWW.

6. References

[1] Joint Task Force on Computing Curricula, "Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering," 2004.

[2] D. A. Kolb, *Experiential learning: experience as the source of learning and development*. Englewood Cliffs, N.J.: Prentice-Hall, 1984.

[3] L. J. Burnell, J. W. Priest, and J. B. Durrett, "Teaching distributed multidisciplinary software development," *IEEE Software,* vol. 19, pp. 86-93, 2002.

[4] E. Ras, R. Carbon, B. Decker, and J. Rech, "Experience management wikis for reflective practice in software capstone projects," *IEEE Transactions on Education,* vol. 50, pp. 312-320, 2007.

[5] B. Decker, E. Ras, J. Rech, P. Jaubert, and M. Rieth, "Wiki-Based Stakeholder Participation in Requirements Engineering," *IEEE Software,* vol. 24, pp. 28-35, 2007.

[6] D. Socha, V. Razmov, and E. Davis, "Teaching Reflective Skills in an Engineering Course," in *Proc. American Society for Engineering Education (ASEE) Annual Conference and Exposition*, 2003.

[7] D. A. Kolb and R. Fry, "Toward an applied theory of experiential learning," in *Theories of Group Process*, C. Cooper, Ed. London: John Wiley, 1975.

[8] D. L. Evans, B. W. McNeill, and G. C. Beakley, "Design in Engineering Education: Past Views of Future Directions.," *Engineering Education,* vol. 80, pp. 517-22, 1990.

[9] K. A. Ericsson, R. T. Krampe, and C. Tesch-Römer, "The Role of Deliberate Practice in the Acquisition of Expert Performance," *Psychological Review,* vol. 100, pp. 363-406, 1993.

[10] E. Ras and S. Weibelzahl, "Embedding Experiences in Micro-didactical Arrangements," in *SEKE 2004*, Banff, Canada, 2004, pp. 55-66.

[11] J. Rech, E. Ras, and B. Decker, "Riki: A System for Knowledge Transfer and Reuse in Software Engineering Projects," in *Open Source for Knowledge and Learning Management: Strategies beyond Tools*, M. Lytras and A. Naeve, Eds.: Idea Group, Inc., 2007.

[12] L. W. Anderson and D. R. Krathwohl, *A taxonomy for learning, teaching, and assessing: a revision of Bloom's taxonomy of educational objectives*, Complete ed. New York: Longman, 2001.

[13] M. Fowler, *Refactoring: Improving the Design of Existing Code*, 1st ed.: Addison-Wesley, 1999.

[14] E. Ras, "Experimental Materials of a Controlled Experiment about the Effect of Learning Spaces on Software Engineering Experience Reuse," Fraunhofer IESE, IESE-Report 002.08/E, Kaiserslautern 2008.

[15] E. A. Karlsson, *Software Reuse: A Holistic Approach*: Wiley and Sons, 1995.

[16] J. Cohen, *Statistical power analysis for the behavioral sciences*. Hillsdale: NJ:Lawrence Erlbaum Associates, 1988.