Using Wikis to support the Net Generation in improving knowledge acquisition in capstone projects

Eric Ras*, Jörg Rech

Fraunhofer Institute for Experimental Software Engineering, Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany

1. Introduction

Professional software engineers are constantly faced with having to cope with ever-changing technologies, along with the need to keep their knowledge up to date. These changes, 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, in practice learning, is less a reaction to ‘being learned’ but more the reaction to a variety of working situations and related problem-solving activities, which requires experience-based learning. Today, most software engineers are from the Baby Boomers generation (born 1946–1964) and from Generation X (born 1965–1980).

The Net Generation (born 1981–1994) differs from the previous generations in terms of commitment, interaction, and learning style and hence puts new challenges upon education and the usage of technology (Prensky, 2001). They have never known a world without computers, the WWW, interactive video games, MP3s, PDAs, and cellular phones. These technologies are internalized by this generation and shape the way they access and use information, as well as how they communicate. Prensky calls them “digital natives,” referring to the fact that they have grown up with technology (Prensky, 2001).

In almost every software engineering curriculum, a capstone course is a mandatory curriculum component. Educational research in this domain has shown that practicing the learned methods and techniques is essential before the students get involved in industrial software development projects. Shaw et al., also stated the importance of practicing what was learned as a core pedagogical principle of software engineering education (Shaw et al., 2006). However, many capstone projects risk overwhelming students because they get overwhelmed with too many new topics, have to understand the different roles and responsibilities assigned, and must 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 (Burnell et al., 2002).

During capstone projects at the University of Kaiserslautern, we, too, made the experience that most students lack practical experience.
for the coordination of development tasks in capstone projects (Decker et al., 2007; Ras et al., 2007). In this platform inspired by the concept of the Experience Factory (Basili et al., 1994), experiences are consolidated by the teachers and fed back into future capstone projects in the following year. Nevertheless, the success of reusing experiences was reduced because of two central problems:

**Bad understanding of reusable artifacts and experience packages:** Understanding is a crucial component of successful reuse and the level of understanding impacts all phases of reuse. However, badly described experiences and their related context leads to bad understandability and applicability of the documented experience, and hence to low perceived information quality. In addition, no adequate support for improving understanding is available in software engineering reuse.

**No explicit support for internalization of knowledge:** Much R&D effort has been spent in the “upward, externalizing” direction, looking for valid experiences that can be formalized, generalized, and tailored. However, the hard part is the “downward, internalizing” direction. Current KM and EM approaches focus mainly on the product of learning and less on the learning processes themselves and on the needs of individuals. Most approaches transfer knowledge by using the “copy model”, i.e., no adaptation of the information and structures takes place when expert knowledge is transferred – it is transferred as documented by experts. Furthermore, novices lack background knowledge and their knowledge is organized differently than the “routine” knowledge of experts. Hence, the copy model does not comply with the structures and processes of human information processing.

A more detailed summary of problems related to understanding and learning from documented experience can be found, for example, in Ras and Weinbzelzahl (2004) and Rech et al. (2007).

This paper first describes the characteristics of the Net Generation and shows which Web 2.0 technologies are compliant with these characteristics and could support the Net Generation in their daily activities (Section 2). Based on this mapping, the selection of the Wiki-based Software Organization Platform (SOP) is motivated. Section 3 presents the core features of the Wiki-based approach for generating so-called learning spaces. The approach supports learning at the workplace through the context-aware generation of learning spaces from previously developed learning content with collaboratively developed documents and experience descriptions. These learning spaces are intended to enhance the internalization of knowledge and are created based on context information from the experience package and information about the current situation. Section 4 explains the design and results of an experiment that was used to evaluate our approach, while Section 5 concludes the paper.

2. Background

Oblinger and Oblinger state that the Net Generation learns by doing and uses computers and the latest in technology in their class work and in their hobbies. They have a wide range of interests, outside their chosen area of study (Oblinger and Oblinger, 2005). Net Generation students are encouraged by the information technology and resources at their disposal. They are social and like to work in teams – the Internet becomes their vehicle of interaction. They are able to perform multitasking in a goal-oriented manner, and follow an immediate "let’s build it" approach (Rickard and Oblinger, 2003).

2.1. Characteristics of the Net Generation

In the following, the main characteristics of the Net Generation are listed:

**C1—Digitally literate:** having grown up with widespread access to technology, the Net Generation is able to intuitively use a variety of information technology devices as well as the Internet (Oblinger and Oblinger, 2005). These technologies are omnipresent and are internalized by the students. The technologies shape the way of how information is created, structured, and disseminated and how social networks are built (see the next section about how the Net Generation perceives technology in general.)

**C2—Connected:** “as long as they’ve been alive, the world has been a connected place, and more than any preceding generation they have seized on the potential of networked media” (Crittenden, 2002). Of course this means that any kind of information is always accessible, that students have the possibility to stay up-to-date, and that they are part of a social network at any time.

**C3—Immediate:** the Net Generation is fast and concentrates more on speed than on accuracy. They multitask and are able to move quickly from one activity to another. The response times are short (e.g., answering to an instance message). They are more used to switch contexts compared to the previous generations.

**C4—Experimental:** most Net Generation learners prefer learning by doing rather by being told what to do. They best learn experientially and prefer the “let’s build it approach” (Rickard and Oblinger, 2003). This implies that they follow a more practical approach and not a theory-based approach to learning and working.

**C5—Communicative:** the Net Generation is very communicative because they like interaction and collaboration. They like to build social networks and work in teams. The Net Generation uses technology extensively to network and socialize (Oblinger and Oblinger, 2005). Open content and open source technologies enable seamless information sharing, collaborative integration of tools, shared creation of media, and collective critiquing and judging.

**C6 — Personalized:** the Net Generation students demand personalized services on the one hand and like to personalize their environment by means of a right set of options on the other hand (e.g., according to interests, personal targets, or preferences such as the presentation of contents, the desired way of navigating through the learning contents, or the learning style) – a one-size-fit-all education will not address their individual preferences and needs. They prefer flexible guidance in education. The personalization is guided by the students and not by the technology.

It can be seen, that the characteristics mesh very closely with the information technologies that increase mobility, their 24/7 availability, and their value as a communication medium. Hence, based on these characteristics, what are the implications for the classroom and the overall learning environment in software engineering education?

It is wrong to assume that Net Generation students prefer mediated interaction and online courses and that they want to use technologies heavily in their education because they grew up with computers and the Internet. Based on the study by Roberts (2005), this is wrong: They like face-to-face social interaction with their peers. While they may use technology in their daily lives, relationships are a driving force in the learning process. Also, Oblinger and
Oblinger state that “it isn’t technology per se that makes learning engaging for the Net Generation; it is the learning activity itself”. Today’s students are experiential learners; therefore, conventional lectures might not be the best learning environment for them. For community-oriented students, peer-to-peer learning opportunities should be preferred to individual activities (Oblinger and Oblinger, 2005).

In reality, the students have high expectations regarding about their education and are willing to express these (Rickard and Oblinger, 2003). Today’s students think of themselves as highly sophisticated customers who have options and who are willing to make choices (Rickard and Oblinger, 2003).

The next section will elaborate the term technology from the perspective of the Net Generation students and will relate (learning) activities to the Net Generation characteristics.

### 2.2. Technologies used by the Net Generation

The Net Generation defines technology broadly. It is not just computers and the Internet, but whatever digital devices or applications that help a student meet his or her needs and support learning activities. A key component of the Net Generation’s definition of technology is customization, or the ability to adapt technology to meet individual needs, rather than vice versa (Roberts, 2005). Net Generation students always think about technologies in combination with the activities they enable.

Today, the Web 2.0 wave has resulted in many Internet-based tools focused on sharing knowledge (Wikipedia), news (Digg.com, true2mors.com), bookmarks (Del.icio.us, spurl, diigo), movies (YouTube), howtos (youteach, howcast), sourcecode (sourceforge), experiences (every blog and forum), etc. The Net Generation stu-

---

### Table 1

Web 2.0 technologies for the Net Generation.

<table>
<thead>
<tr>
<th>Activities (Technology)</th>
<th>Usage scenario</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collaborative authoring (Wikipedia, Wikis)</td>
<td>Wikis are used to edit content on a web server. Everyone (e.g., all project members) can create, extend, modify, or remove the content (e.g., requirements, solutions, technologies, decisions, …)</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>Information sharing/distribution (blogs)</td>
<td>Blogs are used to share information and experiences. One author (e.g., the project manager) creates a blog entry and shares the information (e.g., customer feedback, deadlines, presentations, problems, …) with anyone interested (e.g., registered via RSS)</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Social bookmarking (Del.icio.us, Digg)</td>
<td>Bookmarks are shared by people (e.g., project members) in order to exchange and comment content on other pages (e.g., interesting or conflicting requirements in the Wiki or tutorials on the Internet)</td>
<td>o</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Personal information delivery (Netvibes)</td>
<td>Adaptive portals that aggregate information from freely selected sources (e.g., via RSS), or similar to a dashboard. Multiple sources (e.g., different projects) can be presented, mixed, and filtered</td>
<td></td>
<td></td>
<td>o</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synchronous communication (chats, Skype, cell phones)</td>
<td>Instant synchronous communication channels are used to exchange information in distributed environments (e.g., at distributed locations, when students are at home, with a customer, etc.)</td>
<td></td>
<td></td>
<td>o</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asynchronous communication (emails, micro-blogging)</td>
<td>Asynchronous communication channels are used to exchange and store information for later reuse or to preserve it for other people (e.g., communication with the client that might be needed in later maintenance phases (e.g., decisions of the client))</td>
<td></td>
<td></td>
<td>o</td>
<td>o</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information annotation (tagging, commenting, Diigo)</td>
<td>Tagging or commenting can be used to annotate and classify content in a Wiki or an external site on the Internet. Someone (e.g., the project manager) can classify pages (e.g., the importance of requirements) using tags or comment fixed pages (e.g., negotiated requirements or decisions from the client)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>o</td>
</tr>
</tbody>
</table>

### Table 2

Experience package “Code smell comment”.

<table>
<thead>
<tr>
<th>Attribute (Technology)</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A. Action</strong></td>
<td><strong>Abstract</strong>&lt;br&gt;Comments serve for better communication and explanation of code. It is surprising how often the code is badly commented and that the comments are there because the code is bad. Hence, comments can be substituted by refactoring methods&lt;br&gt;&lt;br&gt;<strong>Problem</strong>&lt;br&gt;Comments are often used to explain bad code. Programmers must add a lot of comment to explain their classes and methods because their naming does not give a hint as to what they intend to do&lt;br&gt;&lt;br&gt;<strong>Solution</strong>&lt;br&gt;The first action in refactoring is to remove the bad code smells. When this is done, many comments become superfluous. In fact, the goal of a routine can often be communicated as well through the routine’s name as it can through a comment. The following refactorings should be used to reduce the comments and improve the code:&lt;br&gt;- When a comment explains a block of code, you can often use the refactoring extract method to pull the block out into a separate method. The comment will often suggest a name for the new method&lt;br&gt;- When a comment explains what a method does (better than the method’s name), use the refactoring rename method using the comment as the basis of the new name&lt;br&gt;When a comment explains preconditions, consider using the refactoring Introduce Assertion to replace the comment with code</td>
</tr>
<tr>
<td><strong>B. Benefit</strong></td>
<td><strong>Effect</strong>&lt;br&gt;Improves communication. May expose duplication&lt;br&gt;&lt;br&gt;<strong>Product context</strong>&lt;br&gt;Digital care giver assistant (DGCA) 1.0&lt;br&gt;&lt;br&gt;<strong>Process context</strong>&lt;br&gt;Agile development process&lt;br&gt;&lt;br&gt;<strong>Project context</strong>&lt;br&gt;Open source Practica 2007&lt;br&gt;&lt;br&gt;<strong>Individual context</strong>&lt;br&gt;Eric Ras&lt;br&gt;&lt;br&gt;<strong>Group context</strong>&lt;br&gt;Team of component “Interaction”&lt;br&gt;&lt;br&gt;<strong>Organization context</strong>&lt;br&gt;Fraunhofer IESE&lt;br&gt;&lt;br&gt;<strong>Customer context</strong>&lt;br&gt;Care center&lt;br&gt;&lt;br&gt;<strong>Software tool context</strong>&lt;br&gt;IDE Eclipse</td>
</tr>
<tr>
<td><strong>D. Description</strong></td>
<td><strong>Detailed description</strong>&lt;br&gt;Code smell comment</td>
</tr>
</tbody>
</table>
Students expect similar tools for their work, hobbies, and entertainment in order to support different (learning) activities.

Several major activities as well as Web 2.0 technologies and systems that can be used for those activities are presented in Table 1. All of these technologies are usable in capstone projects, at least for software engineering. Their support for different Net Generation characteristics is also depicted in Table 1, where the character "○" represent low, "●" medium, and "●●●" high support. For example, Wikis highly support the Net Generation students in the collaborative authoring of software engineering artifacts, which supports their communicative character (C5). However, blogs do not allow students to personalize information sharing or information distribution (C6). The rating in Table 1 was derived by analyzing interviews with five teaching assistants who were knowledgeable about Web 2.0 technologies and the procedures used in capstone projects.

In order to support asynchronous communication and knowledge sharing in a distributed environment, we selected a Wiki that was also used for the collaborative authoring of the software and project documentation within our capstone project. Previous studies in our courses have shown that Wikis are a good medium for content creation and sharing in capstone projects and are very well accepted by students (Ras et al., 2007). Similarly, Gotel et al. also suggest using a Wiki as coordination backbone for student projects (Gotel et al., 2008). In addition to the easy documentation of project tasks, observations, and experiences made, most students are familiar with using Wikis and state that they foster team building and communication.
3. Learning space generation approach

To address the problems stated in the introduction and to reflect the characteristics of the Net Generation, an approach has been developed to produce so-called learning spaces for enhancing the understanding and application of experience packages and knowledge acquisition. These learning spaces enrich experience package with additional information (see Table 2 for a shortened example of an experience package). These are structured according to the A2E template (Rech and Ras, 2007) (i.e., the detailed description (D) has been omitted for a reason).

A learning space is generated by the system when a user accesses a documented experience in the experience base. The generation process enriches the experience package with additional content in order to produce, from a technical point of view, a learning space consisting of a hypertext document with additional 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 (local) situation and the experience package. The learning space is presented by means of Wiki pages within a specialized Wiki.

Two models are essential for the specification and realization of the learning space approach (see Fig. 1): an instructional design model refers to the specification of the overall structure of a learning space by following a specific learning method (e.g., experiential learning, case-based learning, etc.). A learning resource model is dedicated to the specification of the content structures of a learning space. This model implements the concepts of the instructional design model by filling templates with content and by creating physical links between the content artifacts. Both models will be explained briefly in the next two subsections.

3.1. Instructional design model

The instructional design model can be seen as the design of a learning space that follows several instructional design strategies and implements one or several learning methods: (a) providing constructs for specifying an overall learning goal; (b) defining a network of fine-grained learning objectives, which are refined by so-called learning objective templates (LOT); and (c) providing a means for describing these templates, which are sequences of so-called learning activities.

A learning method is a systematic procedure for learning, sustaining, or extending knowledge, skills, and competencies. There are many methods for reaching a learning goal (e.g., reading, writing an abstract, attending a practical course or workshop, learning by teaching, discussing with experts, using e-learning offerings, etc.).

The difference between a learning goal and a learning objective is that usually, learning goals are broad, often imprecise statements of what learners will be able to do when they have completed the learning space. Learning objectives are more specific, have a finer granularity, and are measurable by performing assessments (i.e., through tests, questionnaires). The approach uses Anderson and Krathwohl's taxonomy of educational objectives (Anderson and Krathwohl, 2001), which is a revision of the original taxonomy by Bloom et al. (1956). Hence, regarding the cognitive process dimension, the following six different learning objective types are used in the learning space approach:

**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. Conceptual knowledge provides the basis for understanding. The associated cognitive 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. An exercise can be done by using a well-known procedure that the learner has developed a fairly routinized approach to. A problem is a task for which the learner must locate a procedure to solve the problem. Applying is closely related to procedural knowledge. The associated cognitive 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. Analyzing could be considered as an extension of Understanding and a prelude to Evaluating and Creating. The associated cognitive 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

<table>
<thead>
<tr>
<th>Instructional content elements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning objective</td>
<td>States the selected overall learning objective</td>
</tr>
<tr>
<td>Definition</td>
<td>Provides a definition of a domain concept instance</td>
</tr>
<tr>
<td>Description</td>
<td>Provides a more detailed description of a domain concept instance</td>
</tr>
<tr>
<td>Example</td>
<td>Provides an example of a domain concept instance</td>
</tr>
<tr>
<td>Experience</td>
<td>Provides the detailed description of an experience package</td>
</tr>
<tr>
<td>Scenario</td>
<td>Explains a domain concept instance by means of a practical scenario</td>
</tr>
<tr>
<td>Exercise</td>
<td>Shows an exercise that can be solved by the learner</td>
</tr>
<tr>
<td>Simulation</td>
<td>Illustrates a domain concept instance by a simulation</td>
</tr>
<tr>
<td>Collaborative activity</td>
<td>Provides a contact to a knowledgeable colleague</td>
</tr>
<tr>
<td>Domain relation</td>
<td>Shows a relationship between two domain concepts instances</td>
</tr>
<tr>
<td>Integrated practice</td>
<td>Performs a practice activity in one’s own working environment</td>
</tr>
<tr>
<td>activity</td>
<td></td>
</tr>
<tr>
<td>Situational content elements</td>
<td>Description</td>
</tr>
<tr>
<td>Product</td>
<td>Provides a description of the product of the relevant situation</td>
</tr>
<tr>
<td>Process</td>
<td>Provides a description of the process of the relevant situation</td>
</tr>
<tr>
<td>Project</td>
<td>Provides a description of the project of the relevant situation</td>
</tr>
<tr>
<td>Individual</td>
<td>Provides a description of the individual of the relevant situation</td>
</tr>
<tr>
<td>Group</td>
<td>Provides a description of the group of the relevant situation</td>
</tr>
<tr>
<td>Organization</td>
<td>Provides a description of the organization of the relevant situation</td>
</tr>
<tr>
<td>Customer</td>
<td>Provides a description of the customer of the relevant situation</td>
</tr>
<tr>
<td>Software tool</td>
<td>Provides a description of the software tool of the relevant situation</td>
</tr>
<tr>
<td>Context relation</td>
<td>Shows a relationship between two context concept instances</td>
</tr>
</tbody>
</table>
clearly presented before. The associated cognitive processes are generating, planning, and producing.

The learning space 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.

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” (Socha et al., 2003). Research on experiential learning is based on the work of Kolb (1984), Kolb and Fry (1975), who investigated the learning process related to learning from experiences and whose research has its foundation in the work of Lewin, Dewey, and Piaget. Fig. 1 shows a simple example of a learning space structure template. A global learning goal determines a concrete learning space structure template (e.g., understand code smell; experience package). This template is refined by seven learning objectives templates in this example. Each relates to a specific learning objective (e.g., LOT1: understand code smell; knowledge, or LOT3: understand rename method; process, etc.). Each learning objective consists of a tuple (learning objective type, concept type). The set of possible concept types depends on the specified context model and the domain model (see also Table 3). Each learning objective is refined in a learning activity tree by means of learning objective templates, which are available for each learning objective type/concept type pair. 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 way a global learning goal is refined into learning objectives and activities depends on the learning method to be followed.

For example, the learning objective of LOT3 was “understand rename method; process”, which means that the refactoring process concept “rename method” should be understood by the learner. A possible learning objective template with an activity structure is given in Fig. 2, Tables 4 and 5.

3.2. Learning resource model

The learning resource model can be understood as the realization of the instructional design model. Fig. 1 shows how conceptual artifacts of the instructional design model are mapped to conceptual artifacts of the learning resource model. As can be seen, each learning objective template is realized by a learning page. Such a learning page may contain learning content as well as content from a knowledge management system.

A learning page is composed of so-called content components, which are composed of so-called content elements. Content 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. Content components are units of instruction that contain at least one content element. The difference between a content component and a content element is that a content element has a type, either situational or instructional. Content components realize a learning
objective, respectively, a learning activity. In addition, they can be referenced by another external content component (i.e., a content component of another learning page).

Instructional content elements specify explicit learning content and distinguish between Fundamental, Auxiliary, Orientation, and Resource content elements. Fundamental covers learning content that forms the theory of a specific domain. Since this approach was developed for the software engineering domain, the class Knowledge has been refined by Observation, Experience, Pattern, and Law. Explanation elements serve to provide a deeper understanding. A specific subtype of this category is DomainRelation: these elements present a relationship between two domain concept instances. Illustration elements consist of concrete examples, counter-examples, etc. and can illustrate factual and conceptual knowledge from a practical perspective. Interactivity and Assessment cover content that stimulates the learner to interact with the environment or attend a TestActivity. Content elements of the type Orientation serve the learner by enhancing orientation within a learning space. Resource elements link to external resources (e.g., literature, experts, etc.), or provide cross-references within a learning space (i.e., Pagelink or Component Link).

Situational content elements contain information that describes parts of a situation. By combining them, a situation can be described adequately. In contrast to the content elements, which are stored in the learning content base, situational elements were created in a KM/EM system. Situational elements possess types that exactly correspond to the contextual concepts of the context model.

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 knowledge acquisition in general.

Fig. 3 shows an example of such a generated learning page in the Software Organization Platform (SOP), which was mentioned in the introduction.

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, Germany who took part in a capstone project (18 male students and one female student; eight bachelor students, ten master students, and one Ph.D.

---

### Table 5
Results of the hypothesis tests.

<table>
<thead>
<tr>
<th></th>
<th>One-tailed dependent samples t-test</th>
<th>Wilcoxon test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>df</td>
<td>t</td>
</tr>
<tr>
<td>kad</td>
<td>18</td>
<td>7.168</td>
</tr>
<tr>
<td>kad_cog_remember</td>
<td>15</td>
<td>4.111</td>
</tr>
<tr>
<td>kad_cog_understand</td>
<td>18</td>
<td>4.649</td>
</tr>
<tr>
<td>kad_cog_apply</td>
<td>17</td>
<td>4.227</td>
</tr>
<tr>
<td>kad_cog_analyze</td>
<td>18</td>
<td>3.324</td>
</tr>
<tr>
<td>kad_cog_create</td>
<td>17</td>
<td>2.380</td>
</tr>
</tbody>
</table>

* 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.

---

Fig. 3. Example of a learning page.
student). All of them had a computer science background and were born during the Net Generation period (1981–1994). The evaluation of a briefing questionnaire, which gathered their working and learning preferences, confirmed their Net Generation characteristics. The independent variable (factor) was the set of information provided, i.e., experience packages (EP) or experience packages enriched with learning spaces (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 group and the control group. Because their age did not vary significantly, age was not considered as a confounding factor for the assignment of the students. Statistical comparisons between the experimental and control groups regarding their levels of experience did not reveal any significant differences. In addition, the within-subject design allows reducing the error variance related to the differences among the students (e.g., those related to their level of experience).

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 \( \text{kad} \) – The average overall knowledge acquisition of the experimental group (LSEP) is higher than the average knowledge acquisition of the control group (EP):

\[
H_{1.1} : \mu(\text{kad}_{LSEP}) > \mu(\text{kad}_{EP}); H_{0.1} : \mu(\text{kad}_{LSEP}) \leq \mu(\text{kad}_{EP})
\]

“Average overall knowledge acquisition difference related to the cognitive knowledge dimension \( \text{kad}_{\text{cog}, x} \) – 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:

\[
H_{1.2x} : \mu(\text{kad}_{\text{cog}, x}_{LSEP}) > \mu(\text{kad}_{\text{cog}, x}_{EP}); H_{0.2x} : \mu(\text{kad}_{\text{cog}, x}_{LSEP}) \leq \mu(\text{kad}_{\text{cog}, x}_{EP})
\]

For evaluating the hypotheses, the significance level \( \alpha \) 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 (2001).

The design of the experiment is shown in Fig. 4. 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 one 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 (Flower, 1999). 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 (Ras, 2008) (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 a 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.

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 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 0.665–0.812. This reliability can be stated as high because the test captures quite heterogeneous topics and is related to very different cognitive processes. The scores 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 were 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.

Looking at the means and the relative improvement rates, the subjects of the experimental 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 (Karlsson, 1995). 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 $\gamma$ is a way of describing 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 $\alpha$.

4.4. Discussion

For all dependent variables, the 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 (Cohen, 1988) 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 (Anderson and Krathwohl, 2001). The item analysis ensures that only items with a strong contribution to the overall knowledge acquisition measuring construct were used for analysis. 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 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 the participating software engineering students acquired 204% more knowledge when using learning spaces than with conventional experience descriptions. Based on the data, on our experience from similar studies, and on our knowledge of the German academic and educational system, we assume that the results can be generalized for German computer science students of the Net Generation. However, as no hard data is yet available on the exact composition of the population of the Net Generation, we should be careful to generalize the results for this Net Generation population in general.

Nevertheless, the approach is promising for capstone projects with students of the Net Generation, as it copes with the lack of the students' practical experience by bringing documented experiences, enriched with additional learning content, from more experienced students or practitioners, using Web 2.0 and Internet technologies, into the learning environment. It promises a lightweight solution for collaboratively capturing, organizing, and distributing emergent knowledge and can act as a communication platform within and between Net Generation teams, and, finally, serve as a basis for generating and presenting learning spaces. Because of the social nature of the Net Generation, as well as its preference for experiential learning, interaction is an important issue for universities to support. Students learn more when they interact, with each other, with faculty members, and with material.

The crucial issue is not just how to use technology in an educational setting but also how to understand the Net Generation and how they perceive technology. Since their characteristics are different than those of previous generations, new learning styles emerge that require new ways of how education is presented,
delivered, and assessed. With the appropriate use of Web 2.0 technologies, learning can be made more engaging, social, personalized, and learner-centered—but the uses of information technology are driven by pedagogy, not technology. Universities should not rush into the role of a Web 2.0 technology provider but should use Web 2.0 technologies wisely to cope with the rising demands of Net Generation students.

References


Eric Ras is a senior scientist and project manager at the Fraunhofer Institute for Experimental Software Engineering (IESE) in Kaiserslautern, Germany. He received the B.S. (Vordiplom) and the M.S. (Diplom) in computer science from the University of Kaiserslautern, Germany. From that time, he has worked as a scientist on different public and industrial projects in the domain of empirical software engineering, knowledge- and experience management, e-learning, and document engineering at the Fraunhofer IESE. He has gathered experience in reuse-based learning material production, work-process oriented vocational training methods, and has worked intensively with current e-learning standards and tools. Other research interests are experience management, usage of social software in software engineering, software patterns, the conceptual as well as technical integration of knowledge management and learning management, and ambient intelligent systems for assisted living. He is the author of over 40 journal articles, book chapters, and conference and workshop papers. He is PC member of different workshops and conferences in the domain of software engineering, e-learning, and knowledge management. He is responsible for the international distance learning program Software Engineering for Embedded Systems at the University of Kaiserslautern, Germany.

Joerg Rech is a senior scientist and project manager at the Fraunhofer Institute for Experimental Software Engineering (IESE), an applied research and transfer institute, in Kaiserslautern, Germany. His research mainly concerns software antipatterns, software patterns, defect discovery, software mining, software retrieval, automated software reuse, software analysis, and knowledge management, esp. in the area model-driven software engineering. Joerg Rech authored over 50 international journal articles, book chapters, and refereed conference papers, mainly on software engineering and knowledge management. Additionally, he is a member of the German Computer Society (Gesellschaft für Informatik, GI) and served as a PC member for different workshops and conferences as well as an editor for several books in the domain of software engineering and knowledge management.