PhD Theses in Experimental Software Engineering

Vol. 29

Eric Ras

Learning Spaces:

Automatic Context-Aware Enrichment of Software Engineering Experience

FRAUNHOFER VERLAG

PhD Theses in Experimental Software Engineering
Volume 29

Zugl.: Kaiserslautern, Univ.-Diss.; 2009

Printing:
IRB Mediendienstleistungen des
Fraunhofer-Informationszentrum Raum und Bau IRB, Stuttgart

Printed on acid-free and chlorine-free bleached paper.

ISBN 978-3-8396-0016-0
Fraunhofer-Informationszentrum Raum und Bau IRB
Postfach 80 04 69, 70504 Stuttgart
Nobelstraße 12, 70569 Stuttgart
Telefon +49 711 970 - 25 00
Telefax +49 711 970 - 25 08
E-Mail verlag@fraunhofer.de
URL http://verlag.fraunhofer.de

Learning Spaces: Automatic Context-Aware Enrichment of Software Engineering Experience

Beim Fachbereich Informatik
der Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades

Doktor der Naturwissenschaften (Dr. rer. nat.)

genehmigte Dissertation
von

Dipl.-Technoinform. Eric Ras

Fraunhofer-Institut für Experimentelles Software Engineering
(Fraunhofer IESE)
Kaiserslautern

Berichterstatter:	Prof. Dr. H. Dieter Rombach Prof. Dr. Kurt Schneider
Dekan:	Prof. Dr. Karsten Berns
Tag der Wissenschaftlichen Aussprache:	07.05.2009

D 386

> "Learning Spaces encompass the full range of places in which learning occurs from real to virtual, from classroom to chat room." (M. Brown, 2004)

The term *learning space* is widespread and is used in many different contexts, such as in the name of commercial products, as the synonym for virtual classrooms, or for specific approaches as presented in this work. They all have two things in common: motivate the learner and promote learning as an activity for knowledge construction.

The way how learning takes place today has been shaped by the developments of information technologies. They have brought unique capabilities to learning spaces, such as fostering more interaction between learners, easily accessing a diversity of information, or efficiently creating, tagging, and aggregating of information.

Learning spaces in the context of this work enrich documented software engineering experience with additional information in a didactical manner. They are learner-centered because they reflect the preferences, knowledge, and needs of the software engineer. They are dynamic and situative in terms of their context-aware adaptation. Technically, they consist technically of a hypermedia space of interlinked pages, composed of information from an e-learning system and a knowledge management system in order to support the reuse of experiences on the one hand and knowledge acquisition on the other hand.

Abstract

Software engineering consists of human-based, knowledge-intensive activities in which new situations require new knowledge by the software engineers. An experience factory supports these activities through the collection, analysis, packaging, and dissemination of so-called experiences packages (i.e., knowledge, products, and processes). However, several explorative studies confirm that reuse-based approaches suffer from three problems in practice: bad understanding of reusable artifacts and experience packages in particular; no explicit support for the internalization of knowledge and no compliance with human information processing; and no explicit connection between experience management and technology-enhanced learning approaches.

This work addresses the research question of whether the enrichment of experience packages with additional information (i.e., so-called learning spaces) improves the *understanding* and *application* of an experience package on the one hand and *knowledge acquisition* and *perceived information quality* on the other hand.

The presented learning space approach extends the "project support" activity of the experience factory by automatically generating context-aware learning spaces by merging information from the experience base with learning content. Specified variabilities in generic learning space artifacts support adaptation on the level of structure, content, and presentation to context characteristics.

The main contributions are a reference model consisting of a) a *context model* for describing situations in software engineering, b) a *domain model* for describing the body of knowledge in software engineering, c) a *learning space model* for defining learning spaces on different levels of abstraction (i.e., structure, content, and presentation), d) a *variability model* for defining variabilities in generic artifacts and their resolution, e) a *role model* for implementing the learning space approach in an organization, and f) *techniques* and *tools* for the systematic and automatic on-demand generation of learning spaces (i.e., resolution, adaptation, and presentation). A controlled experiment and a case study provide statistically significant results, which quantify the positive impact of learning spaces upon the understanding and application of experience packages, knowledge acquisition, perceived information quality of experience packages, as well as the use, acceptance, and software ergonomics of the developed tools. A power analysis and effect sizes provide a strong baseline for future evaluations and meta-analysis studies.

Related Publications

The work reported in this thesis has not been submitted in support of an application for another degree at this or any other university. Excerpts of this thesis have been published in books, journals, conference and workshop articles as well as technical reports. An overview of the related publications can be found in the section *Related Publications*.

Acknowledgements

This work could not have been finished without the help of many people who supported me during the last years.

In the first place, I thank Prof. Dr. Dieter Rombach for being my principal advisor, for his guidance and support throughout the whole Ph.D. process, and for giving me the opportunity to perform my Ph.D. work at the Fraunhofer Institute for Experimental Software Engineering (IESE). This provided me with a great environment for applied research. Special thanks also go to Prof. Dr. Kurt Schneider for acting as a referee of my work and to Prof. Dr. Nebel for chairing the Ph.D. committee.

I want to thank my colleagues at Fraunhofer IESE for many fruitful discussions, in particular in the division of competence management. I am grateful to Marcus Ciolkowski, who helped me to plan and design the experiment, and to Björn Decker and Jörg Rech for bringing the Software Organization Platform to life. Thanks to Jörg for his encouraging and supporting way of doing efficient research, for contributions to many publications, and for his support as a refactoring expert. Thanks to Brigitte Göpfert for getting endless lists of articles, to Sonnhild Namingha for correcting my English, and to Stephan Thiel for his layout support. I would like to extend my thanks to the students who evaluated the learning space approach. I appreciate the support of Dimitri Ilin who contributed significantly to the development of the tools.

Thanks also go to external people whose feedback influenced my work, namely Martin Memmel who is a great companion for conducting workshops and tutorials, and for brainstorming about learning approaches. Stefanie Lindstaedt and Andreas Schmidt created perfect opportunities for sharing experiences in the domain of knowledge management and technology-enhanced learning.

Most importantly, I thank my wife Chantal for her patience over years and years of waiting for me while I spent nights on writing many papers and this thesis, and for providing constructive feedback on learning issues. Her love and encouragement in times of hard work helped me to stay on track and strive towards the completion of this thesis. Thanks to our daughter Noa, who always turned my mind to other things and reminded me of the valuable little things beyond doing a Ph.D.

I would like to say apologize to those for whom I did not have enough time during the course of this dissertation.

Table of Contents

List of Figures

List of Tables

List of Definitions

1 Introduction

"The hallmark of professionals is their ability to reuse knowledge and experience to perform their tasks even more efficiently" (Curtis, 1989)

Knowledge is considered an important organizational resource. The ability to learn and to effectively use this resource are two important organizational capabilities. This crucial resource is embedded in organizations through skilled individuals who apply knowledge on a day-to-day basis or stored in knowledge management systems (KMS).

Software engineering is a knowledge-intensive activity

Software engineering, in particular, is de facto a human-based, knowledge-intensive activity. Software engineering follows an experimental paradigm – learning and feedback are natural activities for software development and maintenance (V. R. Basili, Caldiera, & Rombach, 2002). Together with sound methods, techniques, and tools, the quality of software strongly depends on the knowledge and experience brought to the project by its developers. In the past, developers have mostly depended upon tacit knowledge. This resulted in problems when experts left a project and new developers entered. The tacit knowledge was not preserved within the organization, and therefore the steep learning curve for novice developers resulted in a significant reduction in software quality.

Experience management implements the "learning software organization"

Several approaches have shown how parts of this knowledge and the related experiences could be externalized and hence be easier to share with others. Without the reuse of well proven knowledge, for instance in the form of experience descriptions stored in an experience management system, software engineers would have to "recreate" this knowledge on their own or "relearn" it again and again (V. R. Basili & Rombach, 1991). Research on organizational learning has focused on cognitive, social, and organizational impediments to acquiring, sharing, and using knowledge in organizations. In the early 1990s, Senge and Schön began to explore the state of the art and practice of the "Learning Organization" (Schön, 1995; Senge, 1990). In software engineering, the field of *experience management* implements the vision of the "Learning Software Organization". Experience management (EM) is based on the concepts of the Experience Factory (V. R. Basili, Caldiera, & Rombach, 2002), case-based reasoning (Klaus-Dieter Althoff, 2001), and knowledge management (Nonaka & Takeuchi, 1995).

The learning space approach adapts reuse candidate and reuse process

The success of experience-based approaches depends on how well the the reuse process helps the engineer to bridge the gap between the reuse candidate (object to be reused) and the needs (required object). The reuse process itself can be understood as a transformation process between reuse candidate and the required object (V. R. Basili & Rombach, 1991). The learning space approach improves the description of the reuse candidate (i.e., experience) and the reuse process in such a way that better understanding and application (i.e., transformation of the reuse candidate to the needs) is possible.

Structure of this section

Section 1.1 provides a motivation for this thesis and gives basic background information about the fields of experience management and software reuse. Section 1.2 briefly lists the state of the practice and state of the art problems addressed by this work. The main objective of this work is stated in Section 1.3, and the research approach and the related hypotheses are described in Section 1.4. An overview of the developed solution is given in Section 1.5. This chapter ends by listing the contributions to the field of software engineering and experience management, in particular in Section 1.6, and by describing the structure of this thesis in Section 1.7.

1.1 Motivation and Background

The discipline of *software engineering (SE)* was born in 1968 at the NATO conference in Garmisch-Partenkirchen, Germany (Naur & Randell, 1968; Simons, Parmee, & Coward, 2003). At the same conference, the methodical reuse of software components was motivated by Dough McIllroy to improve the quality of large software systems by reusing small, high-quality components (McIllroy, 1968).

The reuse process bridges the gap between reuse candidates and reuse requirements

In the early 1990s Basili and Rombach introduced a model for comprehensive reuse, which has subsequently as a basis for many reuse-oriented approaches (V. R. Basili & Rombach, 1991). They state that a reuse model must be capable of modeling reuse candidates, the reuse requirements, and the reuse process itself. The reuse process helps to bridge the gap between different characteristics of reuse candidates and reuse requirements. The reuse process is the transformation of existing candidates into required objects that satisfy established reuse needs (see Figure 1).

The learning space approach extends the reuse model with respect to the reuse candidates and the reuse process specification. A concrete example can be found in Section 1.5 and the extensions in Section 1.6.

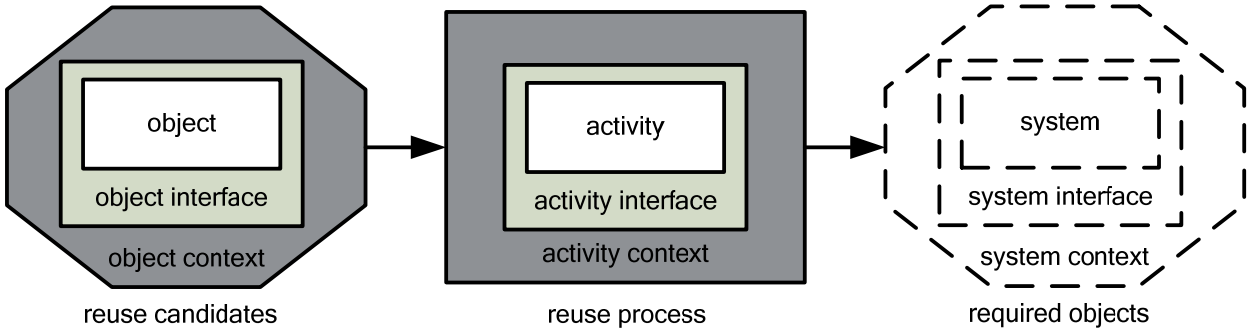


Figure 1 Reuse model of Basili and Rombach (V. R. Basili & Rombach, 1991)

In software engineering, the field of experience management, which subsumes experience reuse, has increasingly gained importance in parallel to software reuse (A. Jedlitschka et al., 2002). In addition, the collection and sharing of explicit experience is one of the three key instruments for implementing experimental software engineering (H. D. Rombach, 2000):

- *Goal-Question-Metric* (GQM) approach for goal-oriented measurement of software products and processes (V. R. Basili, Caldiera, Rombach, & Van Solingen, 2002),
- *Quality Improvement Paradigm* (QIP) for continuous improvement and technology infusion in software engineering (V. R. Basili, Caldiera, & Rombach, 2002; V. R. Basili & Weiss, 1984), and
- *Experience Factory* (EF) organization for enabling the collection of explicit development experiences, their organization for sharing and learning, and their reuse across projects and/or business units (V. R. Basili, Caldiera, & Rombach, 2002)

Experience management is based on the concepts of the Experience Factory

The EF is an infrastructure designed to support experience management (e.g., the reuse of knowledge, products, or processes) in software organizations and is based on the comprehensive reuse model. It supports the collection, pre-processing, analysis, and dissemination of experiences. Today, reuse approaches typically have an organizational unit dedicated to developing, distributing, maintaining, and, often, providing training on reusable assets (Frakes & Kang, 2005). The EF also represents a physical or at least a logical separation of the project and the experience organization as shown in Figure 2. This separation is meant to relieve the project teams from the burden of finding and preserving valuable new experiences that might be reused in later projects. The supporting crew in an EF consists of several roles with different tasks, duties, and rights (Raimund L. Feldmann, Frey, & Mendonca, 2000).

We begin a project by characterizing the environment, setting quantifiable goals for successful project and organization performance and improvement, and choosing the appropriate processes for improvement, as well as supporting methods and tools. We use the EF to search for reusable knowledge in the form of reference architectures, design patterns,

or process models based upon our project context ("Project Support"). During the execution phase ("Execute Process"), the EF is used to retrieve knowledge "on demand". During the project and at the project's end, it is analyzed (e.g., using a post-mortem analysis) in order to extract reusable knowledge (e.g., experience knowledge) that might be useful in other projects ("Analyze"). This knowledge is then packaged in order to be more usable in future projects ("Package").

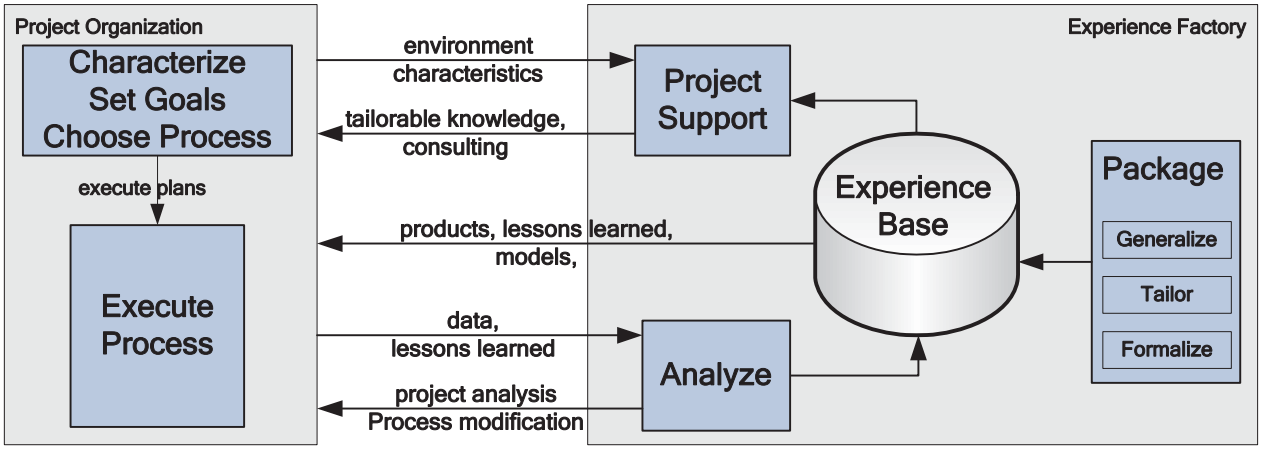


Figure 2 The Experience Factory (V. R. Basili, Caldiera, & Rombach, 2002)

Definition 1 Experience knowledge

In this thesis, *experience knowledge* is defined as:

knowledge that has been gained by acting. It may either result from unprocessed and unreflected events in specific situations or from conscious reflection and interpretation about ongoing things. Experience knowledge is knowledge that can let us act in a practiced and automatic, routine way, or that helps us to judge, select, and apply an appropriate problem solving strategy, method, technique, or tool.

Definition 2 Experience package

In this thesis, an *experience package* is defined as:

an explicit representation of an experience that can be stored, categorized, and disseminated in an organization. It stems from formalizing and generalizing either experience knowledge or experience gained through systematic measurement and improvement. An experience package description contains a problem statement, optionally a proposed solution including the expected benefit/effect when applying it in a new situation, a context description, and additional administrative information.

Experience package is a commonly used term in software engineering, because experience knowledge is "packaged" to make it reusable for future projects.

EMS focus less on long-term competence development

The short innovation cycles in software engineering lead to many learning situations where new knowledge is required to solve new challenges during daily work. Experience management systems (EMS) are one possible solution because they can provide the information in the form of experience descriptions to solve these problems. Nevertheless, their intention is not to provide a learning environment where software engineers can acquire new competencies in a broader sense. Hence, from the individual learner's perspective, the purpose of EMS is short-term problem solving and, much less, long-term competence development.

Learning is experience-based and strongly relies on reflection

The fact that most of our daily learning is experience-based, and that this also applies to software engineering, requires us to take a closer look at the related ongoing learning processes. Educational researchers have argued that one of the key activities of learning from experience is that of reflection (e.g., (John Dewey, 1933; David A. Kolb, 1984) have drawn attention to the role of reflection in Lewin's experiential learning cycle, Schön introduced the concept of the reflective practitioner (Schön, 1990, 1995), and many others have taken the idea of reflection and explored it in the context of theory and practice in experiential learning (see also Section 3.1 for more detailed explanations).

Individual learning is a fundamental part of KM

Rus and Lindvall stated that learning on the individual level is considered to be a fundamental part of KM/EM because employees must internalize (learn) shared knowledge before they can use it to perform specific tasks (I. Rus & Lindvall, 2002). Learning from daily experiences requires reflection about the event that led to the experience and reflection during the application of the experience. However, this reflection process and other learning processes are still not explicitly addressed by current KM/EM approaches, which concentrate more on the product of the learning process (i.e., knowledge) and less on the learning process itself. In the context of experience reuse, not addressing learning processes will result in problems regarding the understanding and application of experience descriptions in new situations.

1.2 Practice and Research Problems

Three state-of-the-practice problems are addressed by this work (see Section 2 for the details):

P-Problem 1 – Bad understanding of reusable artifacts in general and experience packages in particular

Understanding is a crucial component for successful reuse. The level of understanding impacts all phases of reuse. However, badly described experiences and their related context lead to bad understandability and applicability of the documented experience, and low perceived informa-

tion quality. In addition, no adequate support for improving understanding is available in software engineering reuse.

P-Problem 2 – No explicit support for internalization of knowledge and no compliance with human information processing

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. Hence, the information provided is often not structured and presented in a way that it fits human information processing. Furthermore, novices lack background knowledge and their knowledge is organized differently than the "routine" knowledge of experts.

P-Problem 3 – No explicit connection between KM/EM and technology-enhanced learning approaches

The conceptual as well as the technical integration of KM/EM and technology-enhanced learning is still not addressed adequately: The perceived connections between KM/EM and technology-enhanced learning are not operationalized in software engineering, i.e., the integration ideas are rarely implemented in practice, which makes the understanding and application of experience packages, and knowledge acquisition in general, more difficult.

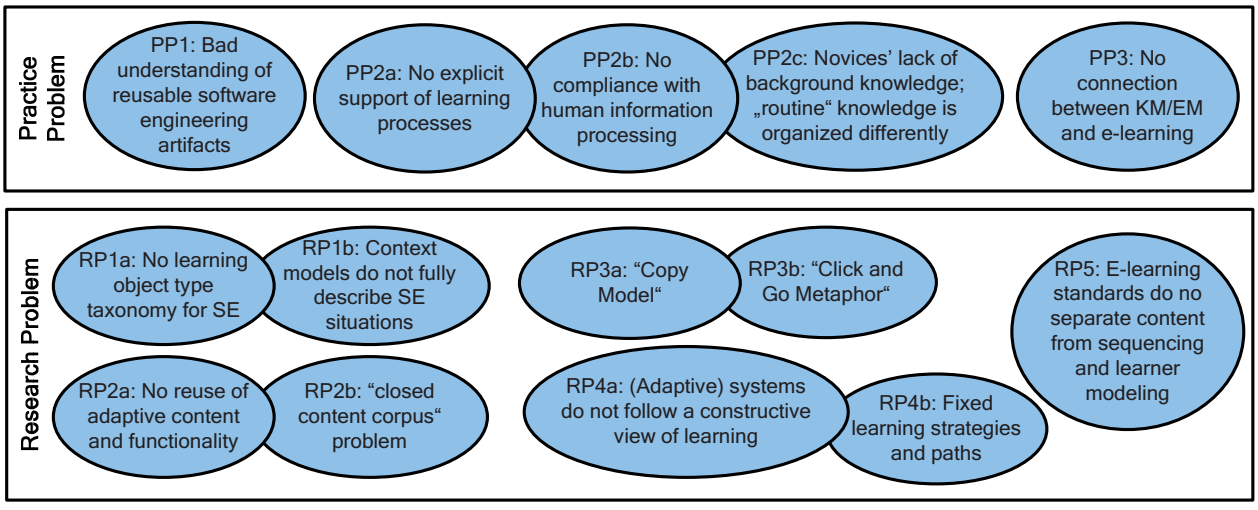


Figure 3 Practice and research problems

In addition to the state-of-the-practice problems, several state-of-the-art problems have been identified (see Section 3 for the details):

R-Problem 1 – No suitable models for learning object types and description of context

Using the IEEE definition of a learning object as a baseline, any developer or researcher should define certain types or categories of learning objects according to his special needs and interests. However, in software engineering, no such learning object type taxonomy exists. Despite the importance of a context description for experience documentation, most of the current experience description schemas do not fully describe environmental characteristics and use semantic relationships between the environmental objects (e.g., product, process, project, organization, customer, etc.)

R-Problem 2 – No reuse of adaptive content/ functionality and no scalability regarding content

Adaptive hypermedia systems (AHS) have a common problem that limits the reusability of their learning resource: The learning resource is intertwined with the logic for generating adaptive content. High cohesion limits the reusability of that learning resource, as the embedded logic often has dependencies on other learning resources. In addition, AHS used a closed set of documents (closed corpus): the documents are fixed at the design stage of the system, and alternations or modifications are hard to process, which results in bad scalability. This problem does not allow opening up the document space or even work in an open environment like the Web.

R-Problem 3 – No content restructuring for learning purposes

Most approaches in software engineering 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. In addition, many systems are based on a hypertext paradigm using the "click&go" metaphor for navigating through an information space. This is a problem because navigating from one fragment to the next based on semantic relationships does not support understanding and learning.

R-Problem 4 – No constructive view of learning and no learning options

First, all Intelligent Tutoring Systems (ITS), many other AHS, and systems in the domain of software engineering are contradictory to the constructivist view of learning. They focus on providing *precise* instructional steps by means of analyzing the learner's state of knowledge in terms of the learner's correct knowledge or misconceptions, whereas the design of constructivist learning environment focuses more on providing *a variety* of learning paths. However, current systems constrain the learner be-

cause of fixed learning strategies and limited possibilities for the learner to investigate topics the system believed to be of no relevance.

R-Problem 5 – No separation of content from sequencing and learner modeling

E-learning reference models and standards do not separate learning content from sequencing and learner modeling. Learner modeling is "hard-wired" into learning objects, i.e., the sequence of learning objects is the same for all learners. As a result, the adaptivity of content is very limited, since it is defined according to a specific learning approach, student type, and a specific set of learning objectives.

1.3 Research Question and Research Objectives of this Thesis

Scope of this work and what it does not cover

As stated before, experience reuse has increasingly gained importance in software engineering. Many success factors have been identified for performing efficient and effective experience reuse: supporting the adaptation of the organizational culture towards organizational learning, convincing management about its advantages, technically building up and maintaining an experience base, developing appropriate description schemata for experience packages, coping with social issues such as motivation and trust, using intelligent similarity-based retrieval mechanisms, etc. This work will concentrate on the reuse activities related to a particular experience package that has been retrieved and selected by the software engineer for reuse. It is not about searching for experience packages, selecting a suitable experience package from a set of candidates, or modifying the experience package. This work develops models, methods, techniques, and tools for supporting the process of *understanding* an experience package and *applying* an experience package in practice. In addition to that, this work investigates the improvement in terms of *knowledge acquisition*, *perceived information quality, use, acceptance,* and *software ergonomics* when the new approach is applied. This work does not investigate whether the learning space approach improves the reuse rate in experience management systems in general, enhances the quality of the products or processes, or decreases effort or time needed. Granted, the more successful application of an experience package in a new situation may lead to these improvements, but this has not been investigated explicitly in this work.

Research Question

This work addresses the research question of whether the enrichment of experience packages with additional information (i.e., so-called learning spaces) improves the *understanding* and *application* of an experience package on the one hand and *knowledge acquisition* and *perceived information quality* on the other hand.

Understanding

Understanding in this context is defined as the understanding of the information in an experience package in the factual and conceptual cognitive knowledge dimension. This means that a reuser is able to remember the facts of an experience package and is able to understand the relationships between those facts.

Application

Application is related to the real application of an experience package in the actual working situation. It requires a higher level of understanding, since it requires that the knowledge acquired on the level of understanding can be applied to practice: The software engineer has developed procedural knowledge and is able to anchor this knowledge with contextual knowledge, i.e., knowing when, where, and why to apply procedural knowledge.

Knowledge acquisition

Knowledge acquisition is more related to long-term competence development, whereas understanding and application are investigated in the context of short-term task performance, i.e., investigating the acquired knowledge for a concrete situation and experience package. Knowledge acquisition investigates whether the software engineer has acquired a deeper understanding of a specific domain and is able to transfer this knowledge to other new situations.

Perceived information quality

Perceived information quality is related to the quality of the information provided in an experience package, respectively learning space. It refers to the usefulness, difficulty, clarity, completeness, etc. of the information.

The three research objectives can be stated as follows (see also Figure 4):

Research objectives

Objective 1: Formally define conceptual models for enriching experience packages with additional information.

Objective 2: Develop a method for the systematic, context-aware adaptation and presentation of learning spaces based on the conceptual models.

Objective 3: Develop a tool for the systematic, context-aware adaptation and presentation of learning spaces based on the conceptual models.

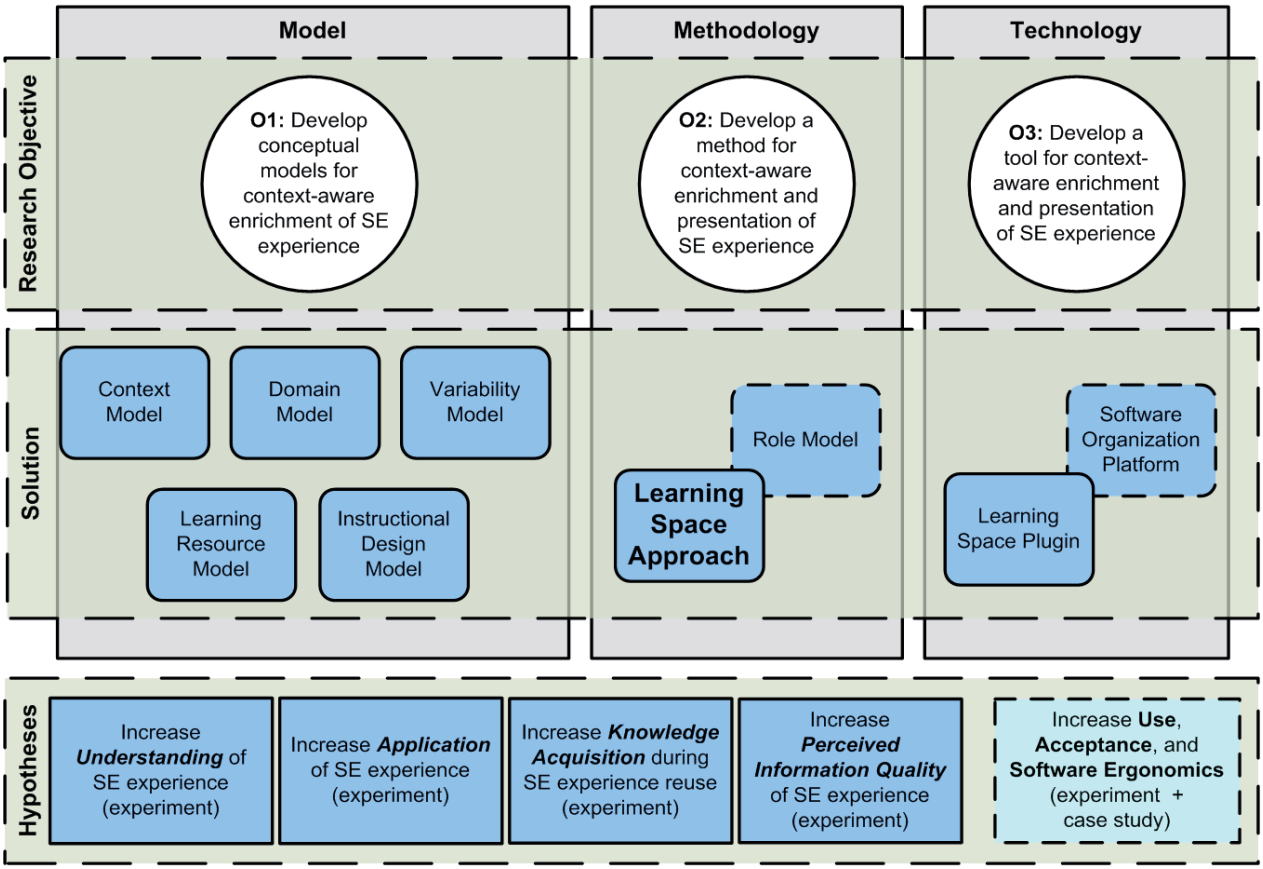


Figure 4 Research objectives, solution, and hypotheses

Hence, in accordance with the GQM approach, the research goal can be stated (V. R. Basili, Caldiera, Rombach et al., 2002) as:

Evaluation goal

Analyze *the effect of learning spaces on experience package reuse* for the purpose of *evaluation* with respect to

- *understanding correctness,*
- *knowledge acquisition differences overall and on the cognitive levels of remembering, understanding, applying, analyzing, and creating*
- *application efficiency, completeness, and accuracy,*
- *perceived information quality, and*
- *use, acceptance, and software ergonomics.*

from the viewpoint of the *researcher* in the context of *a controlled experiment and case study in the domain of experience package reuse at the University of Kaiserslautern*.

The following section describes how this goal can be attained and states the related hypotheses.

1.4 Research Approach and Hypotheses

The research approach is the logical sequence of steps that connect the empirical data to the initial research question and, ultimately, to its conclusions. To realize the learning space approach and attain the goal, the following four steps were necessary:

1st step: Perform literature surveys, market survey, case study

The first step was to conduct several explorative literature surveys, a case study, and a market survey. Based on the case study and the surveys, information about the state of the practice and the state of the art was gathered in several interdisciplinary domains such as knowledge management and experience management, cognitive science and instructional design, technology-enhanced learning in general and adaptive educational hypermedia systems in particular, and, finally, empirical research (see Section 2, Section 3).

2nd step: Develop models and methods

The second step referred to the design of the different models and the development of methods for the learning space approach. A conceptual model was developed to describe the concepts of a learning space. A context model allows first to describe the context of an experience package in a semi-formal way and second, to describe software engineering situations in general. A domain ontology allows the annotation of experience packages and learning resources. A specification was developed to formalize decision models for the adaptive generation of learning spaces. The instructional design of a learning space is expressed by means of specific templates for experiential learning. A learning space generation method was defined for the systematic and automatic generation of learning spaces for experience package enrichment. This step is described in Section 1.

3rd step: Implement models and tool; integrate into SOP

The third step was related to the implementation of the different models and the development of the system itself. In order to implement the models, available specifications and standards (e.g., e-learning specifications) were modified and extended. The other conceptual models were transferred to DTD schemas in order to develop valid XML documents, which were used for generating learning spaces. In order to develop the learning space approach, a new experience management system was developed first, since current solutions did not allow the required technical modifications to the context model developed in this thesis. The EM system was implemented as an extension to the Software Organization Platform (SOP) and allows the documentation and retrieval of experience packages, including their context description. Furthermore, a complete authoring tool for generating learning content was developed. The authoring tool as well as the learning space generation method were implemented as an SOP extension and integrated with the EM system. The resulting tools are described in Section 5.5.2.

4th step: Empirical evaluation

The last step covered the empirical evaluation. A controlled experiment with undergraduate and graduate students and a case study were conducted to investigate the impact of learning spaces upon experience understanding and application, knowledge acquisition, perceived information quality, use, acceptance, and software ergonomics. Five main research hypotheses with their related metrics in parentheses were investigated (see also Figure 4):

Improvement hypotheses

$H_{1.1}$: Using learning spaces during experience reuse leads to an improvement of at least 25% regarding the *understandability* (i.e., *understanding correctness*) of experience packages.

$H_{1.2}$: Using learning spaces during experience reuse leads to an improvement of at least 50% regarding *knowledge acquisition* (i.e., *knowledge acquisition difference in total* and *on five cognitive levels*).

$H_{1.3}$: Using learning spaces during experience reuse leads to an improvement of at least 25% regarding the *application* (i.e., efficiency, completeness, and accuracy) of experience packages.

$H_{1.4}$: Using learning spaces during experience reuse leads to an improvement of at least 25% regarding the *perceived information quality*.

$H_{1.5}$: Using the learning space user interface during experience reuse leads to significantly positive *use, acceptance,* and *software ergonomics* ($p < 0.05$).

Only moderate improvements (i.e., 25%) were expected for understandability, application, and perceived information quality. The reason for this was that a lot of research and development effort has already been put into the development and deployment of experience management approaches in practice. For example, many different templates have been developed and investigated in the past – they evolve over time and have improved a lot. Only a 25% improvement was expected. Nevertheless, since current experience management solutions do not explicitly support learning processes and because EM systems and technology-enhanced learning are still not integrated conceptually and technically, higher improvements (i.e., 50%) were expected for knowledge acquisition. These hypotheses were further refined into statistical hypotheses in Section 7, respectively Section 8. The related measures for the first four hypotheses (used in the experiment) can be found in Section 7.1.2.2 and Section 8 (case study).

1.5 Proposed Solution – An Overview

Learning spaces intend primarily to enhance experience understanding, application, knowledge acquisition, and perceived information quality

To address these problems, an approach was developed to produce so-called learning spaces. A *learning space* is generated by the system when a user accesses an experience package (i.e., experience description) from the database during a project. The generation process enriches the experience package with additional information by following didactical principles. From a technical point of view, a *learning space* consists of a hypertext document with linked learning pages. A learning space follows a specific global learning goal (the learning goal level is selected by the engineer) and is created based on context information about the current situation and context information of the experience package. The learning space is presented by means of Wiki pages (see Figure 5) in a Semantic MediaWiki (MediaWiki, 2009).

In order to generate a learning space, the reuse model of Basili and Rombach was used as a basis and a few characteristics were extended to enable the context-aware generation of learning spaces. The following two tables show example instantiations of the *reuse candidates* and *reuse process* characteristics (V. R. Basili & Rombach, 1991). This example will be used throughout the entire dissertation. Extended characteristics are marked in double-lined boxes:

Table 1 Instantiation of the *Reuse Candidates characteristics*

Comprehensive Reuse Model	*Learning Space Reference Model*
Reuse Candidates Characteristics	
Object	
Name (What is the name of the object?)	- no changes -
"Refactor_Code_Smell_Comment"	"Refactor_Code_Smell_Comment"
Function (What is the function or purpose of the object?)	- no changes -
"Remove Code Smell Comments"	"Remove Code Smell Comments"
Use (How can the object be used?)	- no changes -
"knowledge"	"knowledge"
Type (What is the type of the object?)	- no changes -
"qualitative experience"	"qualitative experience"
Granularity (What is the object's scope?)	- no changes -
"coding stage"	"coding stage"
Representation (How is the object represented?)	- no changes -
"informal description" (see Section 5.3.1 for this experience package, left out here for space reasons)	"learning space"
Object Interface	
Input/Output (What external input/output dependencies does the object have?)	- no changes -
"Java code"	"Java code"
Dependencies (What additional assumptions and dependencies are needed to understand the object?)	as is + selected keywords from the *domain model*
usually informal: "assumes person to be knowledgeable in refactoring and Java programming"	Related domain concepts: code smell comment (knowledge); Java code (product); extract method, introduce assertion, rename method (process)

Comprehensive Reuse Model	*Learning Space Reference Model*
Reuse Candidates Characteristics	
Object Context	
Application domain (What application domain is the object developed for?)	- no changes -
"no specific application domain"	"no specific application domain"
Solution domain (In which environment classes was the object developed?)	as is + selected context instances from the *context model*
agile software development	Related Context Concepts: digital caregiver assistant DCGA 1.0 (product), agile development process (process), open source practica 2007 (project), Eric Ras (individual), team "component interaction" (group), Fraunhofer IESE (organization), IDE Eclipse (software tool)
Object Quality (What quality does the object exibit?)	- no changes -
"not specified"	"not specified"

The extensions are:

- *Object Interface* (Dependencies): instead of an informal description of the dependencies, concepts from the software engineering domain ontology reflect the most important facts used in the experience package.
- *Object Context* (Solution Domain): a more precise description by selecting concrete context instances including their semantic relationships describes the context (e.g., instances from context classes: product, process, project, individual, etc.) where the experience package has been derived/documented.

The standard reuse process consists of the activities *identifying* reuse candidates from the reuse repository, *evaluating* the reuse candidates and *selecting* a candidate, *modifying* the candidate before reuse, if necessary, and finally *integrating* or applying the experience (V. R. Basili & Rombach, 1991). By following a learning space approach a new activity of *generating* a learning space is inserted between the selection and the evaluation process.

In order to further specify the adaptation mechanisms, further subcharacteristics need to be added to *Mechanism*:

- *General Adaptation*: describes whether the adaptation is done by using a decision model or a feature model
- *Adaptation Type*: describes whether the adaptation is done only before runtime (i.e., before the learning space is presented to the user: static) or also during run-time (i.e., when the context changes during the usage of the learning space: dynamic)
- *Adaptation Level*: describes on which level the adaptation takes place (i.e., structure, content, presentation)
- *Adaptation Navigation Techniques*: lists the different adaptation techniques that are used to perform the adaptations on the level of

structure and hence adapt the navigation structure within a learning space (see Section 3.4.1.2 and Section 4.2.4 for examples)

- *Adaptation Presentation Techniques*: lists the different adaptation techniques that are used to perform the adaptations on the level of content and presentation (see Section 3.4.1.2 and Section 4.2.4 for examples)

Table 2 Instantiation of the *Reuse Process characteristics*

Reuse Process Characteristics	
Activity	
Name (What is the name of the activity?)	- no changes -
"Adapt_Generate_Learning_Space"	"Adapt_Generate_Learning_Space"
Function (What is the function performed by the activity?)	- no changes -
"enrich experience package and present a learning space"	"enrich experience package and present a learning space"
Type (What is the type of the activity?)	- no changes -
"generation"	"generation"
Mechanism (How is the activity performed?)	as is + description of adaptation mechanism
"template-based generation via context-aware adaptation"	"General Adaptation: decision model; Adaptation Type: static+dynamic; Adaptation Level: structure, content, presentation; Adaptation Navigation Techniques: Direct Guidance, Link Hiding, Link Generation Adaptation Presentation Techniques: Conditional Text, Fragment Variants, Frame-based Techniques"
Activity Interface	
Input/Output (What external input/output dependencies does the activity have?)	- no changes -
"global learning goal; experience package incl. actual context characteristics, learning space structure template, learning objective template(s), content elements, decision model / learning space"	"global learning goal; experience package incl. actual context characteristics, learning space structure template, learning objective template(s), content elements, decision model / learning space"
Dependencies (What additional assumptions and dependencies are needed to understand the object?)	- no changes -
"performed during coding stage; knowledge about experience package context; knowledge about related domain concepts"	"performed during coding stage; knowledge about experience package context; knowledge about related domain concepts"
Activity Context	
Experience Transfer (What are the support mechanisms for transferring experience across projects?)	- no changes -
"experience base"	"experience base"
Reuse Quality (What is the quality of the reuse activity?)	- no changes -
(see evaluation results in Section 7 & 8)	(see evaluation results in Section 7 & 8)

Next, the basic concepts of a learning space will be briefly explained; however, refer to Section 4 for a detailed description of the concepts and to Section 5 for the underlying approach.

Adaptation and generation of a learning space

Figure 5 illustrates how the reuse model (i.e., reuse candidates and reuse process) can be instantiated in the context of the learning space approach. At the beginning, the learning space is adapted based on context and domain model characteristics of the experience package; it is generated and initially presented to the user. Afterwards, the learning space runs through the states of *presentation*, *interaction*, *context observation*, and *dynamic adaptation*. The system remains in the presentation phase as long as the user does not interact with the learning space. User interaction may either consist of navigation activities within a learning space or changes to the user's situation (e.g., changing software engineering products such as the code). If the interaction requires an adaptive reaction, the context is observed and an adaptation takes place.

Learning spaces are generated based on templates and continuously adapted to the changing context

The generation of a learning space uses an instructional design model (i.e., called *learning space structure template*) with fine-grained learning objectives. It implements the learning process of experiential learning. Before the template is filled with content, each learning objective is refined by means of *learning objective templates* (see Figure 5), which are available for each learning objective/concept type pair (e.g., remember/project, understand/product, apply/process, etc.). For each learning objective template, content elements are retrieved based on the experience package characteristics *solution domain* and *dependencies.*

Content components and content elements are the building blocks of a learning space

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 content elements to form a content 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 component 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 content elements* contain information about the context of the experience package (see bottom right of Figure 5: Learning Page "6"). They have been produced collaboratively by the system users and are stored in a database (e.g., descriptions of projects, individuals, products, processes, organizations, customers, etc.). They should primarily support the understanding of the context of experience packages. *Instructional content elements* are more dedicated to learning topics related to the experience package and to long-term competence development in general. They are stored in the content element database (see Figure 5: rename method, extract method, etc.).

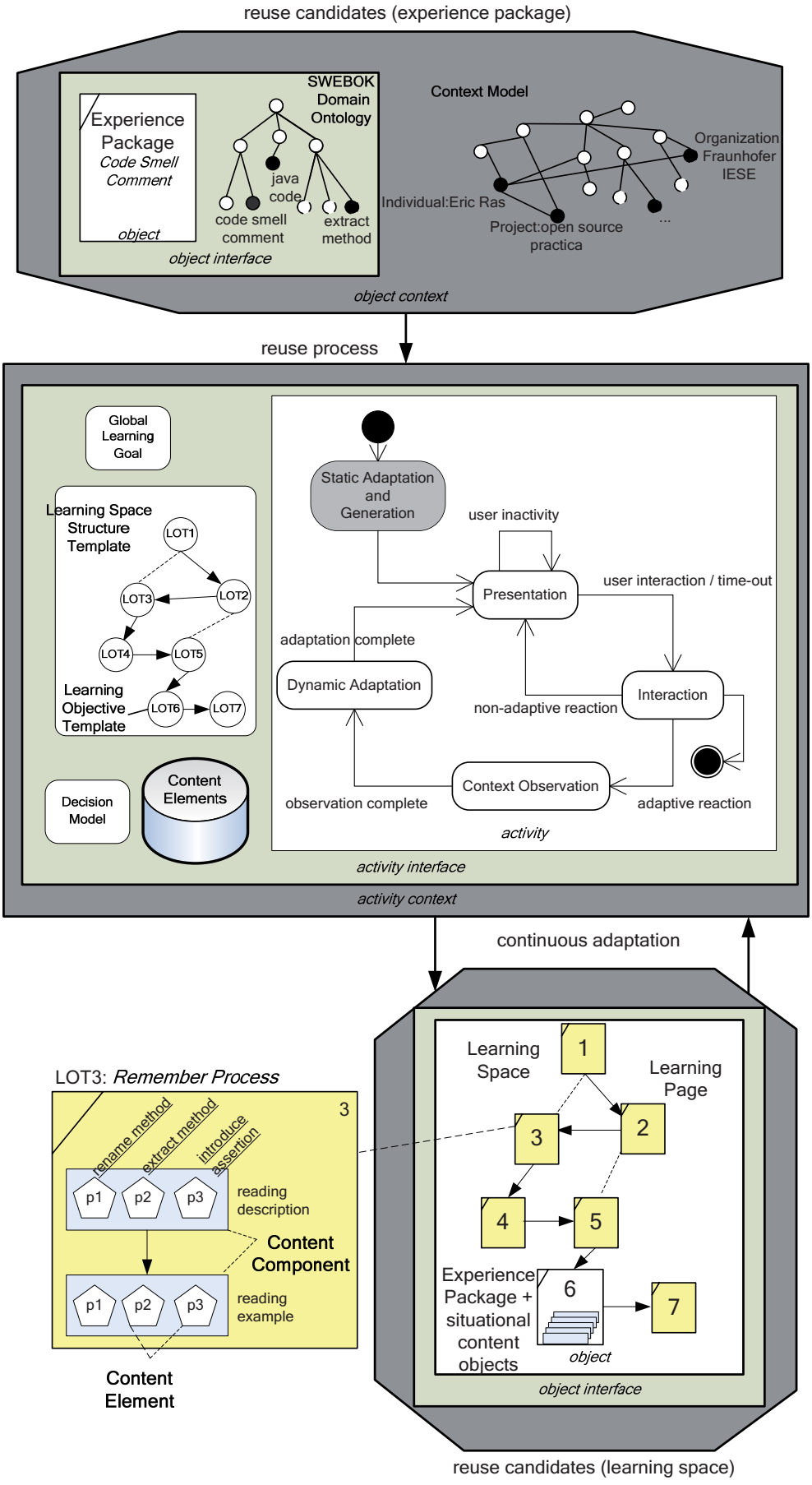


Figure 5 Instantiated reuse model

The learning space approach is part of SOP

The learning space approach was implemented as an extension (kind of a plugin) of the *Software Organization Platform* (SOP) (S. Weber et al., 2008). SOP uses the Semantic MediaWiki (Semantic MediaWiki) as a base platform. SOP intends to support specific software engineering activities such as experience management, requirements engineering, or project management. Hence, by integrating the learning space generation and presentation functionality into SOP, knowledge management and technology-enhanced learning have been merged into one system.

1.6 Contributions of this Thesis

The learning space approach offers contributions to the current state of the art in software engineering as well as that in knowledge management, technology-enhanced learning at the workplace, and adaptive hypermedia approaches in particular (see also Figure 5).

From the perspective of the reuse model, three extensions were necessary to enable context-aware adaptation and generation of learning spaces:

- *Object Interface* (Dependencies) was extended by *Related Domain Concepts*
- *Object Context* (Solution Domain) was extended by *Related Context Concepts*
- *Activity – Mechanisms* was extended to characterize the generation and adaptation activity (i.e., *General Adaptation, Adaptation Type Adaptation Level, Adaptation Navigation Techniques, Adaptation Presentation Techniques)*

From the perspective of the experience factory, the learning space approach extends the "Project Support" activity (see Figure 2). It reuses information from the software organization platform database (SOP DB), such as situational content describing situations in software engineering, learning content, and experience packages. This information is then merged into a learning space; variabilities of the learning space are resolved based on context characteristics; the generic artifacts of a learning space are adapted and finally presented to the user in the project. Another change is that the experience base is not part of the experience factory, but is now a storage medium between the project organization and the experience factory: different types of content are created, stored, and reused from both sides – users have become both content producers and consumers, which supports a more open knowledge sharing community between the project organization and the experience factory.

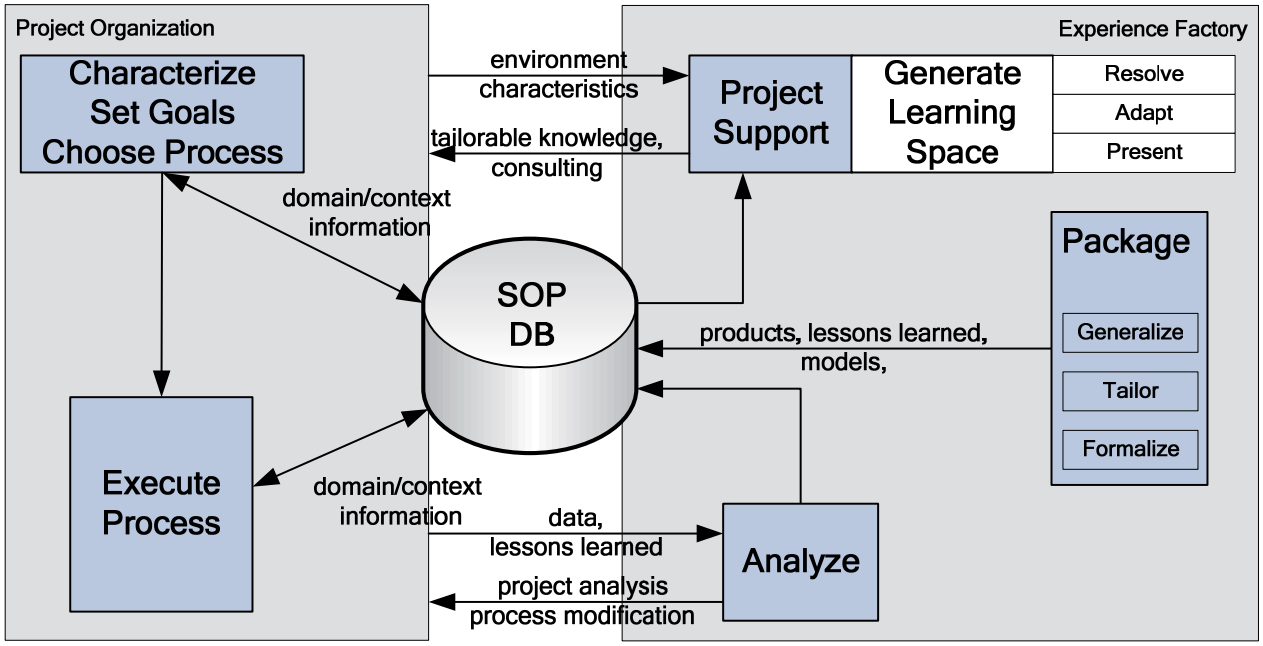


Figure 6 Extension of the experience factory

The main contributions are classified as theoretical, practical, and empirical work.

1. Theoretical work:

- *State of the practice* based on two case studies and a market survey
- *State of the art* based on different literature surveys
- *Context model* for describing situations in software engineering by means of different context concepts and relationships
- *Domain mode*l for describing the body of knowledge in software engineering by means of different domain concepts and relationships
- *Learning space model* for defining learning spaces on different levels of abstraction (i.e., structure, content, and presentation)
- *Variability model* for defining variabilities on different levels of abstraction and their resolution
- *Lifecycle model* for describing the states of a learning space
- *Role model* for implementing the learning space approach in an organization
- *Selection of appropriate learning strategies and methods* for experiential learning in software engineering
- *Techniques for systematic and automatic, on-demand generation of learning spaces* (i.e., resolution, static and dynamic adaptation, presentation of generic artifacts)

2. Practical work:

- Context model available in Wiki syntax for describing software engineering situations

- Domain ontology described in OWL for annotating learning content and experience packages
- DTD schemas for decision model, resolve model, and instructional design templates
- Authoring tool for describing and annotating learning content
- Lightweight experience management system for documenting and retrieving experience packages
- Learning space system for enriching experience packages in a context-aware manner
- Integration of learning content authoring, experience management, and learning space generation

3. Empirical work:

- The empirical evaluations provide statistically significant results, which quantify the impact of learning spaces upon the understanding and application of experience packages as well as the impact on knowledge acquisition and perceived information quality.
- A power analysis and effect sizes provide a strong baseline for future evaluations and meta-analysis studies.

1.7 Structure of the Thesis

The subsequent chapters of this work are organized as follows:

Chapter 2 elaborates the state of the practice and derives the problems by means of a literature survey stating practical examples. The three problems are then further investigated using a market survey and two case studies, which also raise important requirements for developing the learning space approach.

Chapter 3 is concerned with the state of the art. It explains how experts work and how their knowledge is organized in order to motivate why it is difficult to transfer expert knowledge to other software engineers. It provides a definition of the different knowledge types and explains the process of experiential learning. The chapter describes KM and EM approaches and compares them with regard to the knowledge types they support. A section on software reuse explains the comprehensive reuse model and states the advantages of decision models for the adaptation of learning spaces. The chapter ends with several subsections related to technology-enhanced learning in order to state current problems of educational adaptive hypermedia systems, the different types of adaptivity, the relevant e-learing specifications and standards, and finally compares educational systems according to the learning element types they use.

Chapter 4 refers to the learning space approach and describes the two information models of a learning space (i.e., instructional design model and learning resource model), the variability model, the context model, and the domain model. A comparison of the reference models with other reference models of adaptive hypermedia systems is provided at the end of this section and highlights the improvements to existing reference models.

Chapter 5 instantiates the models of Chapter 1 in the context of an experiential learning scenario. A role model describes the different roles involved in the learning space approach and gives examples of their work products (i.e., instantiations/examples of the different models). Afterwards, the techniques related to the static and dynamic adaptation of a learning space as well as the presentation of a learning space are elaborated.

Chapter 6 briefly presents the platform into which the learning space approach has been integrated and lists the open source tools that have been used to develop the models. In addition, the relevant front-end tools developed for this work are explained by means of several screenshots.

Chapter 7 is dedicated to the empirical evaluation of the learning space approach by describing a controlled experiment and its results. It includes the design of the experiment, the data analysis procedure, as well as the very detailed analysis of the gathered data. Due to the fact that experiments related to the didactical augmentation of software engineering experience packages had not been conducted before, this experiment serves as an exploratory evaluation, which can be used as a baseline for future evaluations and developments in this area. Therefore, strong emphasis was placed upon the construction of reliable measurement instruments, the selection of suitable disturbing factors for controlling the experiment, and upon the data analysis itself.

Chapter 8 is concerned with the use, acceptance, and software ergonomics evaluation of the learning space approach: first, as part of the experiment presented in Chapter 7 and second, in a case study mainly focusing on the evaluation of use and acceptance as well as software ergonomics.

Chapter 9 concludes the thesis by summarizing the work and providing a research agenda for future work.

2 State of the Practice

"We can have facts without thinking but we cannot have thinking without facts" (John Dewey)

A systematic literature study of the relevant conference and workshop proceedings, journals, and books was done in order to derive a first set of problems in practice. These findings were strengthened by five workshops, two case studies, and one market survey with 89 companies.

Structure of this chapter

Section 2.1 addresses the problems of reuse in software engineering in practice, Section 2.2 describes problems in knowledge management that are relevant to this thesis, Section 2.3 emphasizes problems related to experience reuse in practice. Section 2.4 lists conceptual as well as technical problems related to the integration of knowledge management and technology-enhanced learning approaches. Finally, Section 2.5 reports the results of two case studies and a market survey.

The different research activities described in this section confirm the three problems stated in Section 1.

2.1 Problems in Software Engineering Reuse

Reuse is more than code reuse

Today, *reuse-oriented software engineering* covers the process of development and evolution of software systems by reusing existing software artifacts. The goal is to develop complex software systems within shorter periods of time or with higher quality by reusing proven, verified, and tested components from internal or external sources. Through systematic reuse of these components and feedback about their application, their internal quality (e.g., reliability) is continuously improved. But reuse of components is only appropriate if the cost of retrieving, understanding and evaluating, selecting, and modifying the component is either lower or if it results in higher quality than that of a component developed from scratch. In the beginning, only the reuse of source code was the focus of reuse-oriented software engineering. Today, the comprehensive reuse of all software artifacts and experiences from the software development process enjoys increased popularity (V. R. Basili & Rombach, 1991). Besides source code artifacts, requirements and design documents, test cases, process and quality models, best practices, etc. are used to support the development and evolution of software systems. These artifacts are collected during development or reengineering processes and are typically stored in special, artifact-specific repositories.

Reusable artifacts need to be developed "for" reuse

Reuse requires that the reused artifacts are "fit for reuse", e.g., that the artifacts are technically developed "for reuse" (i.e., they were developed keeping in mind that they will be reused in future development projects) and represented in an understandable way so that other people who did not create these artifacts are able to reuse them. Biggerstaff stated that skilled staff willing to reuse other people's work as crucial is necessary (Biggerstaff, 1991).

P-Problem 1: Reuse struggles because of bad understanding

However, despite the long tradition of software engineering reuse, systematic reuse is still facing several challenges. These challenges are caused by insufficient support for the reuse steps search, evaluation, and adaptation (Karlsson, 1995). Concerning search, people do not find existing artifacts or do not even start to search due to the effort related to it. Evaluation challenges are mostly caused by either lengthy or insufficient documentation of the artifact found. This often leads to bad understanding and thus difficulties in evaluating the retrieved artifacts. Finally, adaptation refers to the (perceived) effort needed to understand and adapt the artifact in contrast to the effort needed for its initial creation.

Fischer, Henninger, and Redmiles (Fischer, Henninger, & Redmiles, 1991) summarized software reuse problems (Curtis, Krasner, & Iscoe, 1988; Reeves, 1990) as:

- Users do not have well-formed goals and plans
- Users do not know about the existence of the components
- Users do not know how to access components
- Users do not know when to use components
- Users do not understand the results that components produce for them
- Users cannot combine, adapt, and modify components according to their specific needs.

P-Problem 1: Understanding impacts all reuse phases. No adequate support for understanding is available

It can be seen that understanding in general is named as a crucial component for successful reuse by Karlson as well as by Fischer et al. and that is has an impact on all reuse, i.e., from the selection of artifacts to their modification, and on the results these artifacts produce when they are reused. Furthermore, Fischer et al. state that the reason why reuse has not reached its potential is also that no adequate systems for finding, comprehending, and modifying artifacts exist (Fischer et al., 1991).

Frakes and Pole conducted an empirical study about methods for representing reusable software components in the early 1990s. They found that the methods were only moderately helpful in helping the student to understand the components. They motivated that more studies about the effectiveness of the methods for analysis and understanding of components should be conducted (Frakes & Pole, 1994). However, even re-

cently conducted studies (e.g., (Rothenberger, Dooley, Kulkarni, & Nada, 2003) did not investigate the impact of understanding on software reuse in practice.

Understanding is a high cost factor in reuse

In their extensive literature research about reuse programs in practice, Mili et al. (Mili, Mili, Yacoub, & Addy, 1995) emphasize that component/program understanding represents an important part of both the mental effort and the cost factor in reuse (Fischer, 1987) and maintenance (Maiden & Sutcliffe, 1993). In their opinion, component understanding can mean three things:

- understanding what it does,
- understanding how it does it, and
- understanding how to modify it in such a way that it does something a little different.

Existing knowledge about technologies has a higher impact on reuse than software development related aspects

Dusink and Van Katwijk describe the reuse process from different perspectives. From the *engineer's perspective*, they say that for a higher degree of reuse, the reusing engineer's understanding of the reusable artifacts, the process, and the actions to be taken is essential (Dusink & Van Katwijk, 1995). A study from the late 1990s evaluates which factors impact the rate of reuse in practice. An interesting finding was that the hypothesis tests related to the experience level (i.e., knowledge about specific technologies) and the software engineering domain knowledge provided a higher significance ($p < 0.05$) than, for example, software development related aspects such as development effort (p-value < 0.1) (Lee & Litecky, 1997).

Modification (understanding and interface checking) produces most of the reuse costs

The COCOMO II model was developed to meet the need for a cost model that accounts for future software development practices (Center for Software Engineering, 1997). The COCOMO II *size reuse model* describes the non-linear reuse cost function: Only 5% of the costs are related to accessing, selecting, and assimilating the reusable component; the modification to the component to be reused produces most of the costs, i.e., "the cost of understanding the software to be modified, and the relative cost of interface checking (Selby, 1988)."

General reuse education and technology training are necessary

Two principal solutions have been proposed to enhance reuse with respect to understanding. First, Joos mentions education about reuse as a key to gaining acceptance for software reuse in practice (Joos, 1994). A recent study has again confirmed the importance of education in software reuse (Rothenberger et al., 2003). Frakes and Fox found out through an empirical evaluation that education in school and at work influences software reuse (Frakes & Fox, 1995). In addition, developers are more likely to accept a new technology if they are trained in being successful with it (Card & Comer, 1994). Nevertheless, Card and Comer emphasize that reuse training is often misunderstood and more difficult than many managers expect. Second, a very small number of systems

has been developed that intend to support the process of understanding in reuse: Draco, for instance provides domain-specific knowledge (Arango, Baxter, Freeman, & Pidgean, 1985), LaSSIE provides multiple viewpoints of software modules (Devanbu, Brachmann, Selfridge, & Ballard, 1991), Desire-88 supports understanding by recovering design information from the artifacts to be reused (Biggerstaff, 1989), EXPLAINER explains the object, what it does and why (Fischer et al., 1991), and the framework of D'Alessandro et al. provides the specification of generic reusable components (D'Alessandro, Iachini, & Martelli, 1993). Most of the supporting systems had been developed by the early 1990s.

2.2 Problems in Knowledge Management

P-Problem 2a: KM focuses mainly on the product of learning and less on the learning processes

KM systems focus mainly on *organizational learning*, i.e., where learning leads to collecting knowledge for the organization in order to be used by its employees or to modifying the software organization's processes, internal standards, objectives, or strategies. However, Rus and Lindvall state that *individual learning* is considered to be a fundamental part of applied KM because employees must internalize (learn) shared knowledge before they can use it to perform specific tasks (I. Rus & Lindvall, 2002). KM systems make the assumption that the problem of continuous competence development can be partially solved by using intelligent retrieval mechanisms and benefitting from innovative presentations of the retrieval results. KM systems focus mainly on the knowledge (i.e., the product of learning processes), and less on the learning processes themselves and the needs of individuals. Organizations frequently encounter problems in identifying the content, location and use of knowledge. A study showed that 50 to 60 percent of KM deployments fail because organizations did not have a good KM deployment methodology or process, if any at all (Lawton, 2001). The study stated that "next generation" KMS developments should focus on designing KM technologies for people and not make people adapt to KM technologies (Lawton, 2001).

P-Problem 2b: No compliance with human information processing

Designing KM technologies for people, which are suitable for human information processing, means supporting people in their learning processes to ensure that the provided knowledge can be constructed and transferred back into the work process. Enhancing learning means more than sequencing "chunks" of knowledge. It requires an understanding of learning goals and processes, and of the different types of learners and their ways of information processing.

2.3 Problems in Experience Management

Despite the research done in the field of experience management and *Learning Software Organizations* (LSO) (Ruhe & Bomarius, 1999), several

general challenges still exist, which have to be considered when EM approaches are used in practice.

P-Problem 1: Barely described experiences and their related context lead to a bad understandability, applicability, and low perceived information quality

First, Conradi states that in the context of learning organizations externalizing is not the challenge, but internalizing: "Much R&D effort has been spent in the "upward, externalizing" direction, looking for valid experiences that can be analyzed, generalized, and synthesized in the form of improved models and concepts. The hard part is the "downward, internalizing" flow (Conradi, 1999)." Second, Simon describes the knowledge of an engineer as 50,000 chunks, requiring as much as ten years to accumulate (Simon, 1981). Yet, professionals cannot keep all knowledge in their minds, and hence they rely on knowledge in the world (Norman, 1988). Third, Orr describes experience as a socially distributed resource, stored and spread primarily through an oral culture. Interpreting raw measurement data is difficult without extensive routines for classifying data and other context-related information. Written and stored information is barely recognizable to reusers, so the author usually has to be contacted for the context of the experience to be understood (Orr, 1996). Fourth, the quality of the reported experience highly depends on the individual communication skills of the contributor, e.g., the ability to structure the content, to formulate precisely, and to adapt to the potential audience. It is important to realize that not everyone is suited to be an experience communicator. Not everyone has the aptitude and the social skills necessary to transfer their experience pedagogically. This leads to low perceived information quality (Johannson, Hall, & Coquard, 1999). And finally, a general problem is the fact that learning processes have not been explicitly addressed in the context of software engineering reuse (K.-D. Althoff et al., 1999).

In the following, the problems related to expert knowledge and experience transfer are described (a comprehensive description of expert work and experiential learning is given in Section 3.1). Experience is often documented by domain experts. One reason why software engineering knowledge is usually captured from experts is that their knowledge is assumed to be concise, correct, and complete.

So, what makes the transfer of expert knowledge or experiences knowledge difficult?

P-Problem 2c: Novices lack background knowledge and their knowledge is organized differently than the "routine" knowledge of experts

First, from a cognitive science point of view, new knowledge is always related to knowledge existing in human memories. This means that novices might have problems in relating new expert knowledge to their existing "basics". Novices lack software engineering background knowledge and are not able to connect the experience to their knowledge base. Hence, they often misinterpret or even do not understand other people's documented experience. Second, learning is a special case of information acquisition and information storage. The learning process is dependent first on the quality of the information to be learned and second on the cognitive activities of learning. If those activities do not take place because of the problem stated above, the efficiency of information acquisition and storage is decreased. Third, there is not only a quantitative difference between expert and novice knowledge bases, but also a qualitative difference, i.e., in terms of the organization of knowledge (Ericsson, Krampe, & Tesch-Römer, 1993). Cognitive schemata from experts cannot be transferred to the memories of novices. This fact results from a compilation process that is performed when new knowledge is learned: updating or forgetting old knowledge, creating new relationships or rules between knowledge items and aggregating knowledge items, etc. Fourth, asking experts about their knowledge often results in an enumeration of many facts, methods, and principles explained in a complex manner. Experts have forgotten how they learned those knowledge chunks, and they are unable to explain why they choose certain activities and perform them in a certain manner. The applied knowledge is somehow "routine" (Ericsson et al., 1993) (Baumgartner, 2000) and is difficult to externalize. In addition, Adams states that problem solving strategies and so-called thinking strategies are always learned within a specific context of "contents" and are embedded in content-specific schemata, which makes it very difficult to extract them, i.e., to externalize them (Adams, 1989). Finally, transferring past experiences made by others requires more than only contextual knowledge, in particular problem-solving strategies for a specific context and knowing 'when, where, and why' knowledge should be used. It requires a strong anchoring with declarative and procedural knowledge (see Section 3.1 for the types of knowledge).

2.4 Integration of Knowledge Management and Technology-Enhanced Learning

The connection of KM and technology-enhanced learning is still not addressed adequately: An interview-based study showed that perceived connections between these two are not operationalized (Efimova & Swaak, 2002), i.e., integration ideas are rarely implemented in practice. The high potential for synergies between KM and technology-enhanced learning seems obvious given the many interrelationships and dependencies between these two fields. However, the relationships have not yet been fully understood and harnessed. On the one hand, learning is con-

sidered to be a fundamental part of Knowledge Management because employees must internalize, or learn, shared knowledge before they can use it to perform specific tasks. So far, research within KM has addressed learning mostly as part of knowledge sharing processes and focuses on specific forms of informal learning (e.g., learning in a community of practice) or on providing access to learning resources or experts. On the other hand, learning might also benefit from KM technologies. Especially those technologies that focus on the support of technical and organizational components can play an important role with regard to the development of professional technology-enhanced learning systems.

Schmidt states that KM and technology-enhanced learning both serve the same purpose: to facilitate learning and competence development in an organization. However, they follow two different perspectives. KM is related to an organizational perspective, because it addresses the lack of knowledge sharing among members of the organization by encouraging individuals to make their knowledge explicit by creating knowledge chunks that can be stored in repositories for later reuse or by participating in communities of practice; contrary to that, technology-enhanced learning emphasizes an individual perspective, as it focuses on the individual acquisition of new knowledge and the technical means for supporting this construction process (Schmidt, 2005).

P-Problem 3: Perceived connections between KM and technology-enhanced learning are still not implemented adequately

KM addresses learning mostly as part of knowledge sharing processes and focuses on specific forms of informal learning (e.g., learning in a community of practice) or on providing access to learning resources or experts (Efimova & Swaak, 2002). In addition to these interviews, one outcome of a follow-up workshop was that future KM initiatives should shift their focus from knowledge sharing to supporting actual learning from others and applying the experiences of these people (Efimova & Swaak, 2003).

In order to identify and better understand the problems related to the integration of both fields in practice, we conducted a series of four *Learner-oriented Knowledge Management and KM-oriented e-Learning* workshops held in conjunction with famous KM and technology-enhanced learning conferences in Europe (years 2005-2008). Summaries of problems, which are relevant for this work, can be found in (Ras, Memmel, & Weibelzahl, 2005), (M. Memmel, Ras, Weibelzahl, & Burgos, 2006), (Martin Memmel, Ras, Wolpers, & Van Assche, 2007), (Ras, Memmel, Lindstaedt, Ley, & Albert, 2008).

Furthermore, there have been some interesting developments striving to support lifelong learning by integrating technology-enhanced learning systems with experience-based systems in software engineering (K-D. Althoff & Pfahl, 2003). Nevertheless, they are seldom adjusted to the learning demands of individuals, which are very diverse, and they do not

reflect the context of the individual, which is essential to providing context-aware learning services.

2.5 Explorative Studies

Three main problems were identified by means of the literature study and the workshops conducted (in Sections 2.1 through 2.4). In order to a) add strength to what is known through this previous research, b) understand complex issues and related objects, and c) get a first set of requirements for the approach to be developed, further studies were performed: An extensive market survey was conducted to examine contemporary real-life situations in information usage, sharing, and requirements for intelligent assistance approaches (Section 2.5.1). Second, two small case studies with students were conducted in order to perform a more detailed contextual investigation of the previously stated problems (Section 2.5.2). Summaries of the studies are provided in this work. A technical report provides the descriptive statistics and the rationales behind the findings (Ras, 2009a).

2.5.1 'Intelligent Assistance Systems in Software Development' – A Market Survey

A survey with 89 companies was conducted in Germany to shed light on the attitude towards as well as the demand for intelligent assistance in German software organizations (Rech, Ras, & Decker, 2006a). In the following, only findings relevant to the demand for intelligent assistance for *learning* and *competence development* are summarized. Results that focus more on intelligent assistance in general are available in (Rech, Ras, & Decker, 2007a). These are the results relevant for this work:

- The participants evaluated the understandability of the information and its suitability for learning as mediocre (29.2% saying it is good or very good, 50.6% calling it mediocre, and 20.2% considering it bad or very bad) – independent of their position, experience, and company size. → *refers to Problem 1*
- Problems and competence gaps trigger the retrieval of information. Technology-enhanced learning systems should therefore focus on the user's current demands, problems, and knowledge gaps. → *refers to Problem 3*
- 37% of all participants rated expert knowledge as an important information resource. It is interesting that beginners rate expert knowledge as much more relevant (65%) than people with an intermediate experience level (21%), with a statistical significance of 0.004. → *refers to Problem 2*
- Overall, people seem to prefer textual representations for assistance over audio-visual ones – 79.5% prefer short textual descriptions such as tooltips. Likewise, 70.5% of the participants prefer textual assis-

tance in the form of lists and 69.3% would like to have visual assistance in the form of pictures, graphs, or icons. However, animated assistance (e.g., avatars) is rejected by 85.3%, audio assistance by 82.7%, and video was rejected by 59.1%.

- Regarding the reactivity of assistance systems, more than half of the participants prefers reactive systems (53.4%) that are triggered by the user, and more than one third prefers proactive systems (40.9%) that automatically provide learning assistance to the user.
- The participants prefer to see all (52.8%) or at least a filtered selection of potential assistant alternatives. The realization of the proposed assistance should then be triggered by the user (52.8%) and should neither be conducted automatically after a specific period of time nor instantly.
- The participants were asked which learning-specific aspects intelligent assistance should improve (fully applicable=3, partially applicable=2, not applicable=1). Short-term problem solving was rated the highest with an average mean of 2.81.

In summary, the findings showed that there is a high demand and acceptance for unobtrusive, quickly executable, textual, and reactive assistance for short-term problem solving as well as long-term competence development, which enhance especially the understandability of the information provided.

2.5.2 Experience Reuse and Wiki Usage in Software Development – Two Case Studies

The "copy" model was applied during the two case studies, i.e., no changes were made to the experience packages

The first case study was conducted with 14 undergraduate students during a practicum in 2004 that lasted 13 weeks. The students refactored and extended an embedded home automation system in Java by going through all the software engineering development phases. In addition, they were asked to gather process data and to document their experiences. In addition to conventional development tools such as SVN and Eclipse, a Wiki (i.e., TikiWiki (TikiWiki)) with templates was available to document their experiences. The second case study was conducted during an open source practicum in 2006, where 16 graduate students developed a virtual office software for John Deere. This software was developed in order to support distributed software development. The first version of the Software Organization Platform (SOP) with experience package templates was used for experience management. In both practica, experience packages from past practica were offered to the students in order to investigate the reuse of experience packages. The "copy model" was applied for transferring the experience packages (i.e., the information was structured and presented as it has been documented). Two teams developed the software in both practica. In addition to experience reuse, the suitability of Wiki technology was evaluated regarding its support for information exchange and effort savings in soft-

ware development activities, and regarding knowledge acquisition and learning in general. The results of both case studies are discussed jointly in the following (i.e., *cs2004* stands for the first case study and *cs2006* for the second case study). Most questions (marked with "*") uses a five-point Likert-type scale (Likert, 1932) (i.e., "strongly agree" is encoded as "1", "agree" as "2", "neutral" as "3", "disagree" as "4", and "strongly disagree" is encoded as "5"). The other questions were binary questions (marked with "**") and "yes" was encoded as "1" and "no" as "0".

The results are summarized in the following:

- The highest effort savings were observed for experience management for both case studies (cs2004: $M = 2.21$, $SD = .52$; cs2006: $M = 1.79$, $SD = 1.25$). Hence, a Wiki-based system promises to be a good solution for light-weight experience management (*)
- The students rated the suitability of a Wiki for experience management as moderately high (cs2004: $M = 2.13$, $SD = .62$; cs2006: $M = 1.70$, $SD = .79$) (*)
- The Software Organization Platform led to slightly better results regarding the reading of new facts that were unknown before and which would not have been communicated by verbal communication (cs2004: $M = 2.94$, $SD = .93$; cs2006: $M = 2.13$, $SD = .59$) (*)
- When asking whether the students learned about categories of specific facts, the scores for learning were much lower than for reading. This is due to the fact that the system does not explicitly support learning processes (*)
- Most of the new experience packages were created in the categories of tools (cs2004: $M = .63$, $SD = .50$; cs2006: $M = .64$, $SD = .63$) and SE processes (cs2006: $M = .50$, $SD = .65$) (**)
- Technical experience packages related to tools, SE techniques, and processes were also the most prominent ones for reuse (e.g., for tools: cs2004: $M = 1.88$, $SD = .50$; cs2006: $M = 1.96$, $SD = .64$). This led to the decision that the learning space approach should be evaluated in a technical phase close to implementation (*)
- Most of them agree with the fact that the experience packages were reusable by their own team members (cs2004: $M = 2.25$, $SD = .48$; cs2006: $M = 2.50$, $SD = .70$). However, they stated that the experience packages might not be reusable for the other team (cs2004: $M = 3.69$, $SD = .95$; cs2006: $M = 3.39$, $SD = 1.04$) (*)
- The students rated the quality of experience package (i.e., understandability, applicability, etc.) as low (i.e., lower than 3.0) (*)

3 State of the Art

"Praxis is a Greek word that means action with reflection. (Praxis = Experience + Reflection + Action.) In educational situations, we describe, analyze, apply, and then implement our new learning. When we practice a skill, analyze our practice, and then repeat the practice at a higher level, we move practice to praxis. We learn what we're doing" (Marcia L. Conner)

This work is highly interdisciplinary

Structure of this section

The previous sections have described the problems and the research objectives that are addressed by this thesis. In order to develop an appropriate solution, it is absolutely necessary to elaborate the state of the art of relevant fields. Since this work is highly interdisciplinary, additional fields other than software engineering) need to be emphasized to provide a profound understanding of the identified problems and to develop a solution by using and adapting methods, techniques, and tools from the different fields:

- *Professional acting and experiential learning* (Section 3.1): In order to understand why the transfer of experience is so difficult, it is necessary to understand the mental models of experts and the ongoing cognitive processes when humans act and learn from their experiences.
- *Knowledge management and experience management* (Section 3.2): This section provides an overview of existing experience management systems. Then, the role of context in experience management is elaborated, since the explicit description of the context an experience has been documented in plays a crucial role for the generation of learning spaces. Finally, KM systems as well as EM systems are classified according to the explicit knowledge types they make use of.
- *Reuse in software engineering* (Section 3.3): First, the necessary extensions of the reuse model characteristics are motivated. Afterwards, the core concepts and activities of product line engineering are explained because they played an important role during the development of the learning space approach. In order to support the selection of an appropriate product line technique for the learning space approach, the advantages as well as the disadvantages of decision models and feature models are elaborated.
- *Technology-enhanced learning* (Section 3.4): The learning space approach can be classified as a technology-enhanced learning system. Therefore, adaptive hypermedia systems, methods and techniques for

adaptation, and existing e-learning reference models and standards are described. In addition, an overview of classifications for learning resources is given.

Learning processes should be explicitly supported by learning spaces

This chapter shall motivate why learning processes should be explicitly supported by the learning space approach in order to enhance the understanding and application of experience packages. Furthermore, it will shed light on the conceptual and technical challenges as well as on available technologies for developing such an approach.

3.1 Professional Acting and Experiential Learning

This section describes professional acting and to which knowledge and experience types it is related. Further, this section explains the process of experiential learning – which is, de facto, skill and competence development based on experiences. This section is essential to understanding how the important experience knowledge is in software engineering and how experience knowledge is created and transferred between individuals. Insights into the ongoing cognitive processes are necessary in order to later develop the solution, which aims at solving the software engineering problems addressed by this thesis in Section 1.2. After reading this chapter the reader should be able to understand the learning processes if that take place when a software engineer is acting and the kinds of problems that occur when a software engineer reuses an explicit experience that was not described by himself.

3.1.1 How Experts Act

Acting = "Können mithilfe von Wissen"

Neuweg emphasizes that intelligent acting is not necessarily related to defining a goal, developing a solution plan, remembering knowledge, and predicting, but that intelligent acting can also take place in an unconscious, intuitive manner (Neuweg, 2000). He defines acting as 'ability achieved via knowledge' (Germ. „Können mithilfe von Wissen").

Schön as well as Rose justify and appreciate the experience of practitioners (Rose, 1991; Schön, 1995). Rose emphasizes that in working situations where only incomplete information is available and where critical decisions have to be made, appropriate acting is not controlled by formal knowledge but by means of implicit experience knowledge.

Professional acting relies strongly on experts' knowledge and routine

The usefulness of work that relies strongly on experience knowledge was already discovered in production disciplines, e.g., Forschungsverbund "CeA – Computergestützte erfahrungsgeleitete Arbeit" (Martin, 1995): Craftsmen develop new beliefs and ideas based on rememberings and intuitive-associative conceiving of and decision-making in the current situation. New experiences arise when alternatives of acting are tried out exploratively. These insights are not only true for the production domain

but also for software engineering. Programming, for example, is an activity that is strongly guided by explorative acting, which leads to new experiences. Even if professional acting in software engineering should rely on clear goals and processes to be executed, the state of the practice shows that a lot of acting relies on the expert's knowledge and routine and on his ability to adapt his acting to changing situations. Learning based on experiences has become more and more important in recent years because formal learning scenarios can only teach a part of the competencies needed for everyday work (Dehnbostel, 2001). Professionals learn during their work at the workplace in a more informal way – e.g., by experiential learning. Rose depicts that professionals traverse an experience cycle in which they develop ideas about their activity on their own, which they then check and adjust during work (Rose, 1991).

Expert's knowledge is „conditionalized"

The question is how experts act compared to novice practitioners and how their mental models differ. Most experts master complex tasks that require conceptual, procedural knowledge (see Section 3.1.2 for a definition of these terms) as well as a lot of practical experience (Rambow & Bromme, 2000). Experts do not only know at lot about their discipline but their knowledge also reflects a deep understanding of the subject manner and is well organized. In combination, conceptual knowledge and deep understanding (which includes procedure knowledge and metacognitive knowledge, see Section 3.1.2) can help experts as they attempt to apply what they have learned to new situations, thereby overcoming some of the problems of "inert" knowledge (Germ. "Träges Wissen") (Renkl, 1996). An expert should have the ability to recognize meaningful patterns (e.g., generalization, classification) and activate the relevant knowledge of these patterns with little cognitive effort (Bransford, Brown, & Cocking, 1999). Furthermore, expert's knowledge is conditionalized, i.e., they know when and where to use the knowledge (Chi, Feltovich, & Glaser, 1991).

P-Problem 2c: Novice lack background knowledge and their knowledge is organized differently than the "routine" knowledge of experts

Novices lack the "expert" knowledge and the routine of applying this knowledge. From a cognitive point of view, there is a quantitative difference between expert and novice knowledge bases and also a qualitative difference, e.g., the way in which knowledge is organized (Ericsson et al., 1993). Novices lack background knowledge and are not able to connect the experience to their knowledge base. The organization of knowledge at the experience provider's and at the consumer's makes the transfer of knowledge between different levels of expertise extremely difficult. Expert knowledge is, as already mentioned, somehow "routine" because a lot of working processes are performed unconsciously and automatically.

Experts and novices have different problem solving strategies

This is also the reason why specific types of expert knowledge cannot be documented explicitly. In addition, the problem solution approaches of experts and novices differ: Experts solve the problem by starting from the problem; novices must continuously compare the problem with each

performed solution step – they somehow work backwards from the solution idea in order to identify whether the problem can be tackled by the selected solution step. Rambow and Bromme state that expert knowledge is subject to continuous change. This results in a huge amount of experience knowledge that needs to be deployed before it can be applied or made explicit for later reuse. The reason is that experience knowledge is hidden behind the abstract concepts of the domain (Rambow & Bromme, 2000). This phenomenon is called encapsulation (Germ. "Verkapselung").

Even if novices have enough declarative and conceptual knowledge, e.g., as a result of formal education, this knowledge can often not be used in practice (i.e., it is inert knowledge). This is due to the lack of knowledge about when, where, and why to apply this knowledge, or to deficits in recognizing and analyzing working situations (Renkl, 1996). The latter is especially true for those who lack practice. This leads to a loose coupling between declarative and conceptual knowledge.

In order to better understand the previous explanations, the next section provides more details about the different types of knowledge and experience.

3.1.2 Knowledge and Experience Types

The terms knowledge and experience are defined in multiple, more or less formal, and often contradictory ways. Models that define these terms and the processes that transit from one to another differentiate between tacit and implicit knowledge (Nonaka & Takeuchi, 1995; Polanyi, 1966) or between data, information, knowledge, ability, capability, and competence (North, 2002). In addition, knowledge can be conceived at multiple levels, i.e., at the individual level as well as at the group and organizational level.

3.1.2.1 Knowledge and Explicit Knowledge

Knowledge = "Capacity to act"; only exists people's heads

Knowledge can be defined as "a capacity to act" (Neuweg, 2000; Sveiby, 1997). Whether or not knowledge leads to an effective action depends upon people's capacity to interpret the information, generate meaningful options for an action, and perform an action that leads to the desired outcome. There are positions such as that by Stenmark (Stenmark, 2001) that consider the usage of the term "knowledge" for information stored in a computer inappropriate. In this model, tacit knowledge can, in fact, exist only in the heads of people, and explicit knowledge is actually information. However, the terminology used in the theory and practice of information systems (IS) considers knowledge to be information stored together with its context, and I follow this convention throughout this thesis. However, in order to differentiate between

knowledge related to IS and knowledge related to cognitive science, I add the word "explicit" to the IS-related term of knowledge:

Explicit knowledge is information stored in information-based systems (mostly together with contextual data), whereas knowledge is only available in people's heads.

There are many different types of knowledge and even more terms used to describe them (e.g., conceptual knowledge, conditional knowledge, content knowledge, declarative knowledge, discourse knowledge, domain knowledge, episodic knowledge, explicit knowledge, factual knowledge, metacognitive knowledge, prior knowledge, procedural knowledge, semantic knowledge, strategic knowledge, tacit knowledge; for further details, see (P. A. Alexander, Schallert, & Hare, 1991; De Jong & Ferguson-Hessler, 1996; Dochy & Alexander, 1995; Ryle, 1984).

Anderson distinguishes between declarative and procedural knowledge

From among the many different classification schemas for knowledge in cognitive science, two models shall be emphasized here, since they are used as a reference classification for subsequent chapters. The first one is the schema of Anderson, who developed a model of the architecture of human knowledge. He classified knowledge not according to its content but according to its state in the person's long-term memory. Two types of knowledge were defined (J. R. Anderson, 1993; Gagné, 2005):

- *Declarative knowledge* consists of "knowing about" – e.g., facts, impressions, lists, objects and procedures, and "knowing that" certain principles hold. Declarative knowledge is based on concepts that are connected by a set of relations forming a network that models the memory of a person. This leads to the conceptual and theoretical understandings that remain long after many facts are forgotten. For instance, declarative knowledge items in the domain of software engineering might be: a definition of "test case", a listing of defect types, a detailed explanation of key testing principles.
- *Procedural knowledge* consists of "knowing how" to do something, i.e., skills for constructing, connecting, and using declarative knowledge. Learners are doing tasks, such as understanding and processing relationships between items (e.g., facts or objects) and creating new connections between them. Procedural knowledge contains the discrete steps or actions to be taken, and the alternatives available for performing a given task. Procedural knowledge also consists of "if-then" rules that describe when to perform certain actions in a specific situation. These rules are abstract, modular (i.e., they can be combined), goal-oriented, and operate on the basis of declarative knowledge. With sufficient practice, applying the rules of procedural knowledge may become an automatic process, thus allowing the person to perform a task without conscious awareness. For instance, procedural knowledge items in the domain of software engineering might be: knowledge about applying a method for deriving test cases

from a requirements specification, or knowledge about applying a method for classifying defects by choosing the right reading technique during an inspection.

Enn adds contextual (conditional) knowledge to declarative and procedural knowledge

Both declarative and procedural knowledge may be abstract or concrete. The knowledge can be connected to more or less concrete information, which can be described technically, e.g., by semantic networks. Nevertheless, knowledge about situations experienced or about evaluating facts or determining circumstances in given situations cannot be classified as declarative or procedural knowledge. Therefore, a third form of knowledge, conditional or contextual knowledge describing 'when, where and why', has extended the spectrum of knowledge in cognitive science (Enns, 1993). In the context of didactical design, Tennyson and Rasch (Tennyson & Rasch, 1988) defined contextual knowledge as another type of knowledge:

- *Contextual knowledge* consists of "knowing when, where and why" to use or apply declarative or procedural knowledge. Contextual knowledge is created by reflecting on the usage of declarative and procedural knowledge in practice in different contexts. Contextual knowledge enables the individual to be aware of commonalities between situations, and of the appropriateness or applicability of principles or procedures in a new context. Experts possess more contextual knowledge than novices.

Anderson et al. distinguished between factual, conceptual, procedural, and metacognitive knowledge

Anderson and Krathwohl have a similar knowledge classification model, which was developed for classroom instruction and assessment. It provides more details about the subtypes of knowledge compared to the previous model. They separate declarative knowledge into factual and conceptual knowledge. They prefer distinguishing knowledge about discrete concrete elements (i.e., between terms and facts) from larger, more organized aggregation of knowledge (i.e., general concepts, principles, models, or theories). This distinction has also been made in (P. A. Alexander et al., 1991; De Jong & Ferguson-Hessler, 1996).

- *Factual knowledge* refers to the basic elements that experts use in communicating within their discipline, understanding it, and organizing it systematically (L. W. Anderson & Krathwohl, 2001). Factual knowledge can seen as elements and bits of information that can be isolated from their context. Anderson and Krathwohl refined factual knowledge into *knowledge of terminology*, which includes knowledge of verbal and nonverbal labels and symbols (e.g., words, numerals, pictures, which basically represent conventions and agreements in the field), and *knowledge of specific details and elements* (e.g., events, locations, people, dates, sources of information, etc.). These specific facts are basic information used by experts to describe their field and to think about specific problems or topics.
- *Conceptual knowledge* includes knowledge of categories and classifications and the relationship between them and among them – more

complex, organized knowledge forms (L. W. Anderson & Krathwohl, 2001). Several subtypes exist: *knowledge of classifications and categories* is more general and abstract than knowledge of terminology and specific facts. It connects specific elements by links. Anderson and Krathwohl state that "classifications and categories are largely the result of agreement and convenience, whereas knowledge of specific details stems more directly from observation, experimentation and discovery" (p. 49). The second subtype is *knowledge of principles and generalizations.* It is composed of classifications and categories and is used to study phenomena or solve problems in a discipline. This includes abstractions that summarize observations and phenomena that have "the greatest value in describing, predicting, explaining, and determining the most appropriate and relevant action or direction to be taken" (p. 51). The last subtype of conceptual knowledge is *knowledge of theories, models, and structures,* which are the most abstract formulations of conceptual knowledge. The difference to the previous subtypes is that a set of principles and generalizations are related in some way to form a theory, model, or structure.

- *Procedural knowledge* is knowledge about *how* to do things – which might range from routine exercises to solving novel problems (L. W. Anderson & Krathwohl, 2001). It can be expressed as a sequence of steps, collectively known as procedures. It includes knowledge of skills, algorithms, techniques, and methods (P. A. Alexander et al., 1991; De Jong & Ferguson-Hessler, 1996; Dochy & Alexander, 1995). It also includes criteria about *when* to apply certain procedures. Procedural knowledge can be seen as knowledge of different "processes", and factual and conceptual knowledge can be seen as "products" – factual and conceptual knowledge are input and output of the performed procedures that rely on procedural knowledge. The first subtype *knowledge of subject-specific skills and algorithms* refers to procedures (either fixed or flexible ones) that produce fixed results. *Knowledge of subject-specific techniques and methods* is the second subtype of procedural knowledge. It does not necessarily have to lead to a single predetermined answer or solution. This kind of knowledge includes knowledge that is mostly the result of consensus, agreement, or disciplinary norms rather than knowledge as an outcome of observation, experimentation, or discovery. The last subtype is related to *knowledge about when to use appropriate procedures.* This kind of knowledge is an important prelude to a proper use of the procedures themselves. Experts, for example, have for example criteria that help them to make decisions about when, why, and where to apply their knowledge. This category also comprises knowledge about criteria for selecting an appropriate procedure for solving a specific problem (this is related to why).
- *Metacognitive knowledge* is the last knowledge category of Anderson and Krathwohl's knowledge classification schema. It is knowledge about "cognition in general as well as awareness of and knowledge about one's own cognition" (L. W. Anderson & Krathwohl,

2001). When people are aware of their own thinking, they will tend to learn better (Bransford et al., 1999). Metacognitive knowledge is divided into *strategic knowledge* and *knowledge about cognitive tasks, including contextual and conditional knowledge* and *self knowledge.* Strategic knowledge comprises strategies for learning, thinking, and problem solving. It can be used in many different task and subject matters. Especially knowledge about problem solving and thinking is essential for solving ill-defined problems with no predefined solution method (Baron, 2000; Nickerson, Perkins, & Smith, 1985). Contextual and conditional knowledge is knowledge about when and where to use which strategy. Self knowledge includes knowledge of one's strengths and weaknesses in relation to cognition and learning (Flavell, 1992).

The work in this thesis makes use of the classification by Anderson & Krathwohl (i.e., factual, conceptual, procedural, and metacognitive knowledge). This classification subsumes the classification provided by Anderson, Gagné, Enns, Tennyson & Rasch. The next section elaborates the terms *concrete experience* and *experience knowledge*.

3.1.2.2 Concrete Experience and Experience Knowledge

Many definitions and meanings exist regarding the term "experience". Sometimes the word experience is used as a synonym for experience knowledge. The definitions of experience knowledge and experience package used for this work can be found in Section 1.1.

Experience knowledge is developed by acting

Rose describes experience knowledge as knowledge that has been developed through acting and where the consequences of the action have been observed and experienced by oneself (Rose, 1991). Baumgartner describes knowledge for example as a knowledge base of sedimented (Germ. "sedimentierte"), situation-related experiences. He distinguishes first between "routine" knowledge as habitual knowledge that is directly related to the procedures performed and second, concrete experience knowledge. Routine knowledge can be classified into skills (Germ. "Fertigkeiten"), usage knowledge (Germ. "Gebrauchswissen") and recipe knowledge (Germ. "Rezeptwissen"). *Skills* are described here as normal body functions such as speaking and walking. *Usage knowledge* is knowledge related to tasks or activities that are now performed in an automatic manner, and which have been problematic in the past. *Recipe knowledge* is very similar to usage knowledge, but the related actions are less automated and standardized. It refers to *mental routine knowledge* (Germ. "Geistiges Routinewissen") and becomes subject matter specific experience knowledge when it moves away from usage knowledge.

A concrete experience requires different human senses to become experience knowledge

A *concrete experience* can be understood as something that has been experienced in a specific situation. Baumgartner states that a concrete experience requires an interpretation made by using the different human senses before it can be integrated into the knowledge base as *experience knowledge*. During this sedimentation process the experience is idealized, anonymized, and typecasted (Baumgartner, 2000). Baumgartner's statements are based on the work of Schütz (Schütz, 1981), who describes the emergence of the knowledge base not as a result of rational thinking processes, but as the result of sedimentation processes that are related to subjective perceptions, respectively concrete experiences.

Making experience knowledge explicit requires a backward deployment of the aggradation process

Schütz' findings were strengthened by the fact that Polanyi (Polanyi, 1966) and Nonaka and Takeuchi (Nonaka & Takeuchi, 1995), for example, state that the most valuable experiences are tacit, and that it is very hard to express this experience knowledge in an explicit way. Expressing experience knowledge explicitly requires reverse deployment of the sedimentation processes, which is impossible in most cases. Even if the experts are able to externalize their experience knowledge, a transfer to novices is difficult, because the expert knowledge needs restructuring from the expert's perspective to the novice's perspective.

The following section elaborates more on the process of learning through experiences – so-called experiential learning.

3.1.2.3 Experience-based Learning and Experiential Learning

Most learning is experience-based

Most learning is, in fact, experience-based. Whether we hear a lecture, watch a video, read a book, or develop software – our learning is "based" on those experiences. In order to learn effectively from experiences, a learner must be able to perceive information, reflect on how the experience will impact some aspect of his life, compare how the experience fits into his own knowledge base, and be able to think about how this new knowledge offers new ways for to perform in new situations.

In Anglo-American research, so-called *experiential learning* is classified in into two types: first, the way of learning where the learners make experiences in formal learning settings, and second, as "education that occurs as a direct participation in the events of life" (Houle, 1980). The latter refers to learning based on daily situations and the reflection about and interpretation of the experience people make in these situations.

Experience-based learning becomes experiential learning when reflection, abstraction, and transfer of knowledge are done

On the one hand, experiential learning is based more on active "doing" than on passive "being done to". On the other hand, experience-based learning (action alone) becomes "experiential" when elements of reflection, abstraction, and transfer are added to the observed experience. As Rose states, the process of experiential learning is not initiated explicitly (Rose, 1991).

Most of research done in the area of experiential learning is based on the work of Kolb and Fry (D.A. Kolb & Fry, 1975). They investigate the learning processes that take place when people learn from their experiences. Their research is based on the results of Lewin (Lewin, 1951), Dewey (J. Dewey, 1938), and Piaget (Piaget, 1971). Ideally, people could learn effectively from experiences when all four phases of Kolb's Experiential Learning Circle (David A. Kolb, 1984) are passed: a) making a concrete experience, b) observing and reflecting about the occurrence by analyzing the environmental variables and conditions, c) creating new knowledge by forming abstract concepts (e.g., generalizations, principles integrating the observations into sound theories) through comparison of the experience with the existing experience knowledge, and, finally, d) applying and testing these concepts in new situations and checking whether the results meet the expectations (see Figure 7). This active experimentation allows the learner to try out what he has learned in new and more complex situations and for problem-solving and decision-making. This knowledge is used as a basis for acquiring more concrete experiences, which in turn forms the basis for new knowledge and competence development. Due to the sedimentation of experiences, individual do not depend on older experiences, since these are merged and refined into a new and deeper understanding (see previous section).

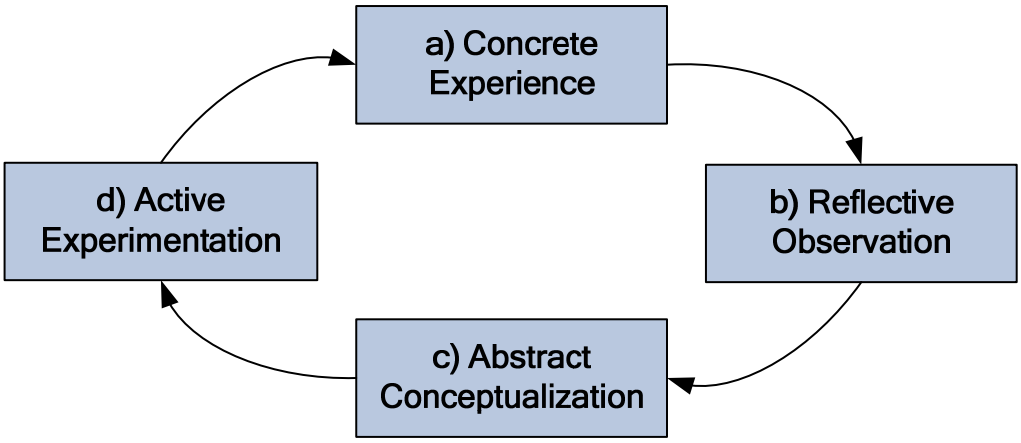


Figure 7 Experiential learning cycle of Kolb (D.A. Kolb & Fry, 1975)

Reflection is essential in order to learn from experiences

One cornerstone of the learning cycle is the process of reflection, since reflection is the most important activity for to learning from experiences (David A. Kolb, 1984). Kolb describes the ability to reflect as a prerequisite for the success of experiential learning. By reflecting about the experiences made, a learner gains new insights and competencies. Piaget

(Piaget, 1976) distinguishes between *accommodation* (Germ. "Reflektierung") and assimilation (Germ. "Reflexion"). Accommodation is a constructive reflective activity and supports the development of new concepts and schemas that arise from environmental experiences, whereas assimilation integrates new experiences into existing knowledge schemas. Both types of reflection interact.

3.2 Organizational Learning and Related Approaches

Single-loop, double-loop, and deutero learning

Agyris and Schön define organizational learning as "the detection and correction of error". In this context, they distinguish between three different levels of learning: single-loop-learning, double-loop-learning, and deutero learning (Argyris & Schön, 1978).

- *Single-loop learning* – Single-loop learning occurs when errors are detected and corrected and organizations continue with their present policies, rules, and goals. Dodgson said that single-loop learning can be compared to activities that add to the knowledge base or organization-specific competencies or routines without altering the fundamental nature of the organization's activities (Dodgson, 1993). Single-loop learning has also been referred to as "Lower-Level Learning" (Fiol & Lyles, 1985), "Adaptive Learning" or "Coping" (Senge, 1990), and "Non Strategic Learning" (Mason, 1993).
- *Double-loop learning* – Double-loop learning happens when, in addition to the detection and correction of errors, the organization reflects on and modifies its existing standards, procedures, policies, rules, and goals. Double-loop learning leads to changes in the organization's knowledge base or in organization-specific competencies or routines (Dodgson, 1993). Double-loop learning is also called "Higher-Level Learning" (Fiol & Lyles, 1985), "Generative Learning" or "Learning to Expand an Organization's Capabilities" (Senge, 1990), and "Strategic Learning" (Mason, 1993).
- *Deutero learning* – Deutero learning is about knowing how to perform single-loop learning and double-loop learning. The first two forms of learning will not happen if the organizations are not aware that learning must occur.

Double-loop learning and deutero learning are concerned with the why and how to change the organization, while single-loop learning is concerned with accepting changes without reflecting about and questioning underlying assumptions and core beliefs. Knowledge and experience management systems have implemented these types of learning in organizations in the early 1990s up to now.

Knowledge management

Knowledge management is the systematic and organizationally specified process of acquiring, organizing, and communicating human knowledge so that other employees may make use of it in order to be more effective

and productive in their work (Alavi & Leidner, 1999). Knowledge management systems are tools for supporting the management of knowledge and are manifested in a variety of implementations (Davenport, De Long, & Beers, 1998). These include, for example, document repositories, expertise databases, discussion lists, and context-specific retrieval.

Subsection 3.2.1 provides a short summary of the most relevant experience factories. Section 3.2.2 motivates the importance of context information and reports on how context has been described in different experience management approaches. Finally, a classification of knowledge and experience management approaches according to the explicit knowledge types they support is given in Section 3.2.3. These knowledge types provide first insights into what could be used as *situational learning components* in learning spaces in order to enhance experience package reuse in Section 1.

3.2.1 Experience Factories in Industry and Research

Knowledge management systems (KMS) and experience management systems (EMS) have been developed to address the problem of knowledge loss and to improve knowledge sharing in general. The *Experience Factory* concept was introduced to support software development (see also Section 2.3). This led to the development of experience-based information systems from the late 1990s on and continues until the present day. A lot of research has been done on packaging experience by building informal and formal models, on performing measures of various software processes, products, and other forms of knowledge (Victor R. Basili, Daskalantonakis, & Yacobellis, 1994), as well as on the customization of EM systems to organizational needs (Tautz, 2001). Furthermore, the *Learning Software Organization* (LSO) (Ruhe & Bomarius, 1999) researches methods and techniques for the management, elicitation, and adaptation of reusable artifacts from SE projects.

Experience factories incoroporate quantitative and qualitative experience packages

Most of the well-known Experience Factories (EF) focus on quantitative experiences packages, i.e., new experiences are gathered through systematic and goal-oriented measurement. Nevertheless, most of the following examples also incorporate qualitative experience packages in their infrastructures. Some of them consist of shorter descriptions, other experiences are documented by reports, trainings, or even by means of tools for cost estimation.

Examples of Experience Factories

The NASA Software Engineering Laboratory (SEL) (Victor R. Basili, Caldiera, McGarry et al., 1992) covers all software engineering activities and stores experience in the form of product-, process-, tool-, relationship-, management-, and data packages. Daimler-Benz uses an EF that focuses on requirements engineering, formal reviews, risk management, software inspections, and quality assurance (Houdek, Schneider, & Wie-

ser, 1998; Schneider, Hunnius, & Basili, 2002). The related LID approach (stands for *Light-Weight Documentation of Experiences*) is a software process improvement approach. It consists of a process and templates to create reusable material for different kinds of users (Schneider, 2000). Hughes Aircraft (Humphrey, 1991) stores their – mostly quantitative – experiences for software process improvement in the form of written reports, recommendations, and trainings. They have focused on experience regarding cost estimation, error data, and schedule performance. Information about technologies used in projects is stored in a technology transfer database. The software process improvement (SPI) initiative at Raytheon (Haley, 1996) emphasizes defect data and defect density, requirements, software cost and scheduling, program design languages, design and code inspections, design reviews, coding standards, unit testing and software integration, regression testing, and prototyping. The Experience Factory COIN at Fraunhofer IESE (Andreas Jedlitschka, Althoff, Decker, Hartkopf, & Nick, 2001) is designed to capture qualitative experience of past software engineering research projects, i.e., experiences regarding the application of specific technologies, process models, or research approaches, etc. Motorola's EF (Victor R. Basili et al., 1994) focuses especially on the domain of reviews and testing, e.g., in terms of software quality defects, software functionality, effectiveness of defect removal activities, error and defect trends, etc. The *Component Factory* (CF) as a specialization of the EF is concerned with the reuse of software artifacts (V. R. Basili, Caldiera, & Cantone, 1992) and builds the framework in which further analysis and retrieval techniques are embedded.

A lesson learned is knowledge or understanding gained by experience

Another type of system that is not based on the EF concept is the *Lessons Learned System* (LLS) (R. Weber, Aha, & Becerra-Fernandez, 2001). Amongst the definitions for lessons learned, the most complete definition as stated by Weber et al. (R. Weber, Aha et al., 2001) p.3 is: "A lesson learned is knowledge or understanding gained by experience. The experience may be positive, as in a successful test or mission, or negative, as in a mishap or failure.

R-Problem 2b: Transferring knowledge is done by using the "copy model"

The standard method used in practice for transferring knowledge from experts to novices is the "copy model": Expert knowledge is considered as learning material and is transferred directly to the learner by using an appropriate medium. Even if this model is commonly applied in many different educational contexts, in KM systems as well as in EM systems, it does not comply with the structures and processes of human information processing (J. R. Anderson & Graf, 2001).

3.2.2 The Role of Context in Experience Management

In computing and related subjects, context is also widely used, often with different meanings. Context has a specific meaning in AI (Lieberman & Selker, 2000) and natural language processing (NLP)

(Lenat, 1998), which differs to a great extent from the notion of context in operating systems and programming languages. The understanding of context in design and user interfaces engineering is again quite different.

What is context?

The context of a reusable artifact can be seen as all the information that does not describe the artifact itself. A context description includes, for example, knowledge about social, physical, historical, and other circumstances where actions or events happen (resp. where the reusable artifact is applied or used) (Brézillon, 1999). In fact, this knowledge is not part of the actions to be executed or the events that occur, but it will constrain the execution of an action and the interpretation of an event. When the core of the artifact/experience represents the "how", the context characterizes the "where", "when", "why", "what", "by whom", etc. the artifact was created/used in. In general, the artifact represents the data, and all environmental metadata (i.e., data about data) is context. Context explains the environment the knowledge was created in and is a way of giving knowledge meaning and focus. The more understandable and focused it is, the most effectively it can be captured and reused in a given situation (Araujo, Santoro, Brézillon, Borges, & Rosa, 2004)

Most common definition of context

The most widely used definition is the one of Dey. Dey defines context as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." (A. K. Dey, 2001)

The following definitions are taken from different online dictionaries.

- *Context is defined as the interrelated conditions in which something exists or occurs* (Merriam-Webster, http://www.merriam-webster.com/).
- *Context is defined as 1: discourse that surrounds a language unit and helps to determine its interpretation [syn: linguistic context, context of use] 2: the set of facts or circumstances that surround a situation or event; "the historical context.* (WordNet®1.6, http://www.cogsci.princeton.edu/~wn/)
- *Context is that which surrounds, and gives meaning to, something else.* (The Free On-line Dictionary of Computing, http://foldoc.doc.ic.ac.uk/foldoc/)
- Synonyms of context*: Circumstance, situation, phase, position, posture, attitude, place, point; terms; regime; footing, standing, status, occasion, surroundings, environment, location, dependence. (Online Thesaurus, http://www.thesaurus.com)*

Context cannot be defined alone, without considering the object it is related to

Looking at the definitions of context (such as those described above) and more detailed explanations of context in PhD theses in the field (Schilit, 1995), (A. Dey, 2000), (Pascoe, 2001), (Mitchell, 2002), it can be observed that the notion of context evolves with the work that is carried out and that it can be seen as a multidimensional space, which by nature cannot be determined in its entirely. In addition, context cannot be defined alone, without considering the object it is related to. It is external to this object and is restricted to the meaningful elements with regard to this object.

The acquisition and formalization of context information plays a central role in experience documentation and reuse. Therefore, the last part of this section focuses on context and describes how current experience package schemas support the formalization of contextual information. Due to the fact that the description of context depends on the artifact it surrounds, and that the experience management approaches differ in terms of the artifacts they store, it is impossible to compare the context dimensions.

Context information is part of the reuse model

Basili and Rombach state that, besides object and interface information, context information is essential in their reuse model (V. R. Basili & Rombach, 1991). Tautz says that the characterization of a reuse object must include context information (Tautz, 2001). In addition to these approaches, much work has been done on context description for specific types of artifacts. For example, Damiani et al. use software descriptors and fuzzy weights to define the behavior of software components in order to increase the understanding for reengineering or dynamic domain modeling (Damiani, Fugini, & Bellettini, 1999).

Examples of context information

So far, context information in software reuse has been used mostly for classification and retrieval purposes. One aspect that has been addressed frequently is the information that is necessary to help the developer select and reuse appropriate methods, techniques, and technologies in a certain software engineering context. For example, technology domain models describe the class of context situations (e.g., kinds of software projects) in which a software engineering technology can be applied successfully. They reference the technology, its application task, its goal, as well as the viewpoint and the overall environment for which the domain model has been derived (A. Birk, 2001). Birk and Kröschel introduce so-called *Technology Experience Packages* (TEP). A TEP describes a technology, specifies to which process (e.g., software design or measurement) it can be applied, which product quality (e.g., robustness) of which product type (e.g., information systems) can be achieved by applying it, and in which context situation this application and the quality impact of the technology can be expected (A. Birk & Kröschel, 1999). Another interesting approach is to use the description of a problem situation as context description of a solution that can be reused (Andreas Birk & Tautz, 1998). The combination of information about solution-problem

pairs, context, and object information has been used for several lessons-learned systems. Houdek and Kempter show how the quantitative results of a goal-oriented measurement program can be packaged in so-called *quality patterns* (Houdek & Kempter, 1997). A quality pattern has a problem-solution pair at its core, which is enhanced by further information (i.e., abstract, context, example, explanation, related experience, etc).

Other context models that emphasize place less emphasis on software quality aspects and more on the collaborative characteristic of software development focus on the description of the group context. Rosa et al. believe that the important elements for the composition of the group context are divided into five information categories: people, scheduled tasks, relationships between people and tasks, environment where the tasks are accomplished, and concluded tasks (Rosa, Borges, & Santoro, 2003). Araujo et al. base their work on Rosa's by using the following context facets: individual, roles, team, task, project, organization, client, product, software engineering domain, client business domain knowledge (Araujo et al., 2004). However, they do not provide any details about those facets.

R-Problem 1b: Current context models do not cover all environmental dimensions

Despite the importance of a context description for experience documentation, most of the current experience description schemas do not contain environmental data at all, or just cover a few categories mentioned previously: The Quality Pattern (Houdek & Kempter, 1997) template contains a context field, but does not provide a more detailed structure for describing the context; the same is true for an experience template for product line software development processes (Kamel, M., & Sorenson, 2001). A good overview of different experience repository schemas can be found in (Lindvall, Frey, Costa, & Tesoriero, 2001). The Q-Labs context attribute for instance, covers only customer-, project-, and technical-area data; the Fraunhofer Maryland experience repository schema is more detailed, and also states information about competencies, process, customers, and documents, but does not cover a product category; Birk's TEP schema (A. Birk, 2001) covers almost all context categories but limits the value range of the attributes by pre-defined values from taxonomies.

In the following section, different knowledge and experience management systems from different domains are compared according to the types of explicit knowledge they address. This overview is then utilized in Section 1 as a basis for deriving a context vector and types of *situational learning components*.

3.2.3 Classification of Knowledge/Experience Management Approaches

Definition of six explicit knowledge types

There exist many classification schemas for explicit knowledge (i.e., information stored in information-based systems, see Section 3.1.2.1). Mittelmann, for example, distinguishes between product knowledge, process knowledge, leadership knowledge, environmental knowledge, social knowledge, and expert knowledge (Mittelmann, 2005). Basili and Rombach consider product, process, and expert knowledge (V. R. Basili & Rombach, 1991). Based on those classifications and the classfications of (Langenbacher, 2002), (Mödritscher, 2005), (Probst, Raub, & Romhardt, 1998), and (Steiger, 2000), the following explicit knowledge types will be used to classify KM and EM approaches:

- *Product knowledge* is related to a specific product. This might be an end product (e.g., software delivered to a customer) or an intermediate development artifact that was created to develop the end product.
- *Process knowledge* refers to information about processes, workflows, activities, and tasks that are understood and applied in a company. It describes the methods and techniques used to produce and consume products.
- *Expert knowledge* consists of information stored in the system that elaborates more complex decisions about selected methods and techniques, models, principles, laws, patterns, etc. that have been derived based on observations of systematic measurement and improvement of products and processes.
- *Leadership knowledge* documents knowledge about guidance and direction of work in an organization. It is about inspirational motivation and intellectual stimulation of individuals or groups of individuals, about adapting and influencing the organizational culture, and about the strategies applied.
- *Market knowledge* refers to information about customers, competitors, market situations, potentials, demands, etc.
- *Organizational knowledge* elaborates information about organizational structures, policies and rules, organizational development, and behaviors to be followed inside and outside the organization.

Table 3 shows which explicit knowledge types are mainly supported by the different approaches ("+" stands for basic support; "++" stands for strong support).

None of the approaches covers all knowledge types

It can be seen that none of the approaches covers all specific knowledge types, since they were developed for different purposes. The system from software engineering (mostly based on the Experience Factory paradigm) covers mainly the types of product, process, and expert knowledge. KMS from other domains focus more on leadership or organizational knowledge and put less emphasis on product and process

knowledge. Only a few consider explicit knowledge about customers, markets trends and potentials, and similar issues.

Table 3 Classification of KM and EM approaches

Approaches	Product	Process	Expert	Leadership	Market	Organization
(V. R. Basili, Caldiera, & Cantone, 1992)	++		++			
(Victor R. Basili et al., 1994)	++	++	++			
(Chang, 2005)	+	++				
(Dewe, 2001)	+	+		+		+
(Falbo, 2004)	+	+	+	+		+
(Fraunhofer IFF, 2006)	+		++		++	
(Fröming, 2005)		+	+	+		+
(Goeken, 2005)	+	+				
(Gronau, 2006)		+		+		+
(Guretzky, 2002)	+	+	+	+		+
(Haas, 2006)	+	+	+			
(Haley, 1996)	++	+				
(Houdek et al., 1998)	+	+	+			
(Humphrey, 1991)		++	++			+
(Andreas Jedlitschka et al., 2001)	+	++				++
(Langenbacher, 2002)	+	+	+	+		+
(Lenk, 2002)	+	+		+		+
(Mittelmann, 2005)	+	+	+	+		+
(Mödritscher, 2005)	+	+	+	+		+
(Natali, 2002)		+	+	+		+
(Group, 2004)	+		+		+	
(I. Rus, Lindvall, M., Sinha, S., 2001)	++		++	+		+
(I. Rus & Lindvall, 2002)	+		++	+		+
(Steiger, 2000)	+	+	+	+		+

3.3 Reuse in Software Engineering

The first section explains which categories of the comprehensive reuse model (V. R. Basili & Rombach, 1991) need to be adapted in order to support a context-aware enrichment of experience packages. The second section highlights the most valuable concepts of product line engineering for realizing adaptivity in learning spaces.

3.3.1 Comprehensive Reuse Model

As stated earlier in the introduction and also in Section 3.2.2, Basili and Rombach introduced a model for comprehensive reuse, which has been used as a basis for many reuse-oriented approaches in the future (V. R. Basili & Rombach, 1991), for example in the domain of maintenance, where the activity of understanding the reuse candidates and the required objects (i.e., specification) is crucial (H. D. Rombach, 1991). Figure 8 shows the detailed reuse model.

The characteristics of the reuse model are insufficient for adaptive reuse activities

Nevertheless, the reuse process does not allow specifying adaptive reuse activities such as the adaptation and generation process for the learning space approach. Therefore, extensions are necessary to the characteristic *mechanism* of the reuse process in order to specify different variations of adaptations. In addition, an important input for the adaptation process is context information and a more precise description of the domain concepts related to a specific experience package. This information is needed to adapt the learning space to context information of the experience package (i.e., reuse candidate) and second to know which concepts must be understood in order to reuse the experience package. Hence, extensions to the characteristics *dependencies* and *solution domain* are necessary.

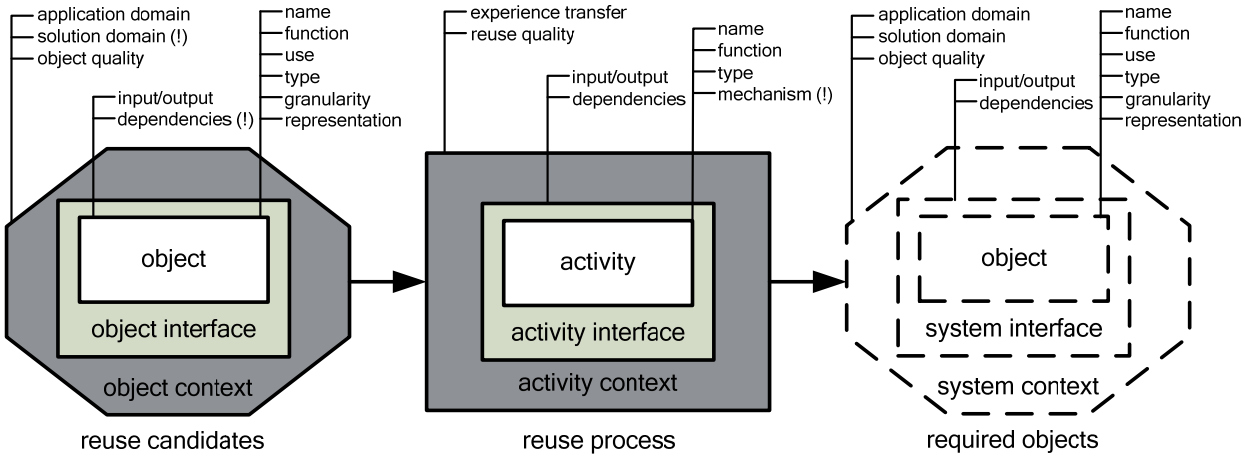


Figure 8 Reuse model of Basili and Rombach (V. R. Basili & Rombach, 1991)

3.3.2 Product Line Engineering Activities

Product line engineering is a software engineering reuse approach and helps software organizations to identify commonalities and variabilities in their lines of product variants. The product line engineering approach uses the knowledge about these common characteristics to define the skeleton or common core of a reuse infrastructure (Muthig, 2002). Clements and Nothrop define a software product line as "a set of software-intensive systems sharing a common, managed set of features that satisfy needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way" (Clements & Northrop, 2001).

Family engineering and application engineering are the two main activities

Product line engineering is related to domain engineering. Coplien defines *domain engineering* as "a software design that focuses on the abstraction of a business (a domain) with the intent of reusing design and artifacts" (Coplien, 1998). In contrast to domain engineering, product line engineering focuses on a set of individual products to be developed. Domain engineering is a domain analysis-based activity that analyzes an application domain completely and builds a reference model for

software products in this domain. One part of domain engineering is the activity of *family engineering,* which develops and fills a so-called artifact base belonging to the reuse infrastructure for software systems of the domain under consideration. Product line information describes the characteristics of the set of systems under consideration, as well as the related variabilities and commonalities (Muthig, 2002).

Product family

A *product family* is related to the solution domain and reflects a series of products that can be built from a common set of building blocks – the so-called *assets*. Assets are product line artifacts and are stored in the artifact base. A *product line* is related to the problem domain and refers to a summation of products that share a common set of characteristics and specific customer needs of a specific domain or market (Becker, 2004).

Application Engineering

Application engineering is the activity that produces a concrete product and uses the product line information for customizing the system to the customer's needs. During the instantiation of the generic assets (i.e., variable assets) the variabilities are resolved by using either a decision model or a feature model (see next section). The instantiation can resolve decisions from different phases, e.g., starting from the requirements phase and ending with the implementation phase. The result of the instantiation is a *resolve model* describing how the generic assets are transformed into regular assets in which no variability is left. Additional steps are usually necessary to develop functionalities that were not covered by the product line reuse infrastructure.

Product line engineering usually resolves variabilities during development time

The difference between software development with and without product line engineering is that a set of software systems is systematically investigated regarding their commonalities in order to build a reuse infrastructure for supporting later reuse rather than developing system after system from scratch without any systematic reuse. A very important aspect is the time when variabilities are resolved. In product line engineering variabilities are defined that are not related to run-time adaptivity, i.e., system-based adaptations based on input data. Muthig states that product line engineering focuses on capturing variabilities amongst systems that are resolved during development and not during execution (Muthig, 2002). Nevertheless, he said that it is a strategic decision whether variability is specified as choices during development time, or whether it should be implemented into a system for resolution at run-time.

3.3.3 Modeling Variability – Feature Models and Decision Models

Decision models and feature models are the most common techniques for variability modeling

Generic software products differ from non-generic products by possessing variabilities. Van Gurp et al. define variability as the possibility to modify and adapt a system (Van Gurp, Bosch, & Svahnberg, 2000). Variability is modeled and defined in the related assets, which makes them *generic assets*. These variabilities can be resolved during different phases of development, i.e., variabilities may be related to requirements, design, and code. In product line engineering the instantiation of these software products is always done during development and not during run-time (see previous section). The location of these variabilities is called variation point. Variabilities determine what kind of product with different behavior can be instantiated. Several techniques exist for describing variabilities in product lines. The two most common techniques for describing and managing variabilities are feature models and decision models.

Feature model

Feature models are created in the feature-oriented domain analysis method (FODA) (Kang, Cohen, Hess, Novak, & Peterson, 1990). This method defines a feature as "an aspect, quality, or characteristics of a system that is visible to the end-user". Feature models are described by means of feature trees. Features can be *mandatory*, i.e., the system must support the feature; *optional*, i.e., a feature can be supported, but this is not necessary; *alternative*, i.e., a feature must be supported, but it can be chosen how it is supported and exactly one alternative has to be chosen; *or*, a feature must be supported, but more than one alternative can be chosen (Czarnecki & Eisenecker, 2000). Van Gurp et al., for example, use variable features (Van Gurp, Bosch, & Svahnberg, 2001). Composition rules define the dependencies between features (i.e., *require rule* and *mutually exclusive rule*). Trapp states that current feature models do not support variabilities that have to be resolved during run-time and therefore adds *dynamic requires rules* (Trapp, 2005).

Decision model

Decision models are an alternative to feature models and are based on the synthesis approach of Kasunic (Kasunic, 1992). Ziadi et al. describe that "a decision model represents the set of relevant decisions and their impacts that are needed to identify one single product in a product line" (Ziadi, Jézéquel, & Fondement, 2003). A decision model is "a product line artifact that captures relationships among variation points in a set of generic artifacts" (Muthig, 2002). Hence, they are a means for structuring variation points and consist of decisions with questions. Decisions may influence other decisions. This information is described by using so-called resolution constraints. There are three types of resolution constraints: complete, partial, and exclude. *Complete resolution* means that a decision completely resolves a variation point. *Partial resolution* resolves a part of a variation point, i.e., other decisions are necessary to resolve the variation point completely. *Exclude resolution* constraints exclude other decisions, i.e., those decisions become obsolete (Muthig,

2002). Bayer et al. state that a "decision model captures variability in a product line in terms of open decisions and possible resolutions. In a decision model instance, all decisions are resolved. As variabilities in generic workproducts refer to these decisions, a decision model instance defines a specific instance of each generic workproduct and thus specifies a particular product line member (Bayer, Flege, & Gacek, 2000). A resolution model consists of all decisions and the answers to their questions and is a product line artifact. The activity of *decision modeling* documents and maintains the relationships between variation points and the characteristics of a product family.

Decision models can be resolved bottom-up or top-down

Decision models are usually created iteratively in a bottom-up way and form a hierarchy of decisions (other forms are possible; this depends on the types of dependencies used). Nevertheless, resolving these decision models can be done in two ways: either bottom-up or top-down. Bottom-up means that that all variation points of the generic product are resolved one by one by means of the decision model. Top-down means that for each identified product characteristic, the decision model is resolved from the top to the leaf nodes (i.e., simple decisions). A resolve model is the result of both options. The KobrA (Komponentenbasierte Anwendungsentwicklung) method developed at Fraunhofer IESE follows the product line approach and uses decision models for variability management (Atkinson et al., 2002). Decisions are built bottom up, i.e., for each variability, a *simple decision* is created, which belongs to the lowest level of abstraction. These decisions are combined to build so-called *complex decisions*, which is an activity of decision modeling. During application engineering, the decision model in KobrA is resolved in a top-down manner.

3.3.4 Feature Models versus Decision Models

Decision models support traceability - feature models do not

Decision models and feature models are essential product line artifacts of a product line infrastructure. However, there are some major differences, advantages, and drawbacks, which will be important when choosing the appropriate techniques for the learning space approach during a later stage of this thesis. Feature models, for example, document variabilities as well as commonalities, whereas decision models only cover variabilities. An advantage of decision models is that they ensure traceability along the different software development phases (Dhungana, 2006; Dhungana, Kepler, Rabiser, & Grunbacher, 2007). Traceability is realized because each decision refers to all product line artifacts on different levels of abstraction (i.e., requirements, design, and implementation). However, feature modeling does not support traceability because no mechanism is available for connecting the feature models of the different abstraction layers.

Decision models are not limited to a hierarchical structure as feature models are

Pech says that decision models allow modeling very complex dependencies between decisions, whereas "feature models only allow modeling dependencies for features and their sub-features, so the dependencies are restricted to mutual exclusion of hierarchical composition dependencies" (Pech, 2007). In addition, Kim et al. mention that decision models allow defining n:m relationships between decisions – meaning that simple tree hierarchies can be easily exceeded (Kim, Kim, Shin, & Baik, 2006). However, inconsistencies in the decision model may occur when modeling is not done carefully (Pech, 2007).

3.4 Technology-Enhanced Learning

Continuous competence development is essential to keep track with the requirements of today's work environments. This trend can be observed especially in the Information and Communication Technologies sector with its increasing flood of information, rapid deterioration and hence ageing of knowledge, as well as the continuously changing requirements for problem understanding and solving. As a result, these facts require lifelong learning at the workplace to remain competitive in the information society.

In industrial training settings, learning objectives mostly correspond to concrete, well-defined job-related skills, specific tasks to be done, or problems to be solved. The delivered learning material must suit the current situation that the software developer is currently in. The situation changes over time while the software developer is performing his work.

Learning delivered online is called e-learning or technology-enhanced learning

Learning delivered online, refers to as *e-learning* or *technology-enhanced learning*, gives learners a self-controlled learning experience via a computer. However, most conventional technology-enhanced learning offers still suffer from one size fits all (Conklin, 1987), whereby each learner receives an identical learning experience. These learning offerings have had high drop out rates as learners become increasingly dissatisfied with courses that do not engage them (Frankola, 2001). Such high drop out rates and lack of learner satisfaction are due to the fact that most current e-learning offerings deliver the same static content to all learners, irrespective of their prior knowledge, experience, preferences, or goals.

Learning becomes bite-sized

Conventional learning systems leave no space for dynamic selection and sequencing of learning resources (Brusilovsky & Vassileva, 2003). New types of learning services and mechanisms need to be developed and provided because "learning becomes fragmented and bite-sized (Bonar, 1988)." As learning has become more learner-centered, technology-enhanced learning and related technologies have also become increasingly personalized (Sharples, 2000). So-called adaptive hypermedia systems have the potential to address current educational barriers of technology-enhanced learning by allowing learning to be tailored to specific

user needs and preferences. In order to realize on-demand and just-in-time learning systems, learning is now based on small information chunks (e.g., so-called learning objects) rather than on conventional formal courses and seminars that cover broader topic areas.

Section 3.4.1 will describe the state of the art in adaptive educational systems. Section 3.4.2 focuses more on conventional technology-enhanced learning approaches. It explains how current standards and specifications support adaptivity, how instructional design is done with learning objects, and provides a classification of learning objects. Section 3.4.3 first defines instructional design and describes its history and provides second a classification of learning object types.

3.4.1 Intelligent Tutoring Systems and Adaptive Hypermedia Systems

R-Problem 4b: Intelligent tutoring systems constrain learners because of fixed learning strategies

In the domain of education, so-called intelligent tutoring systems (ITS) were the first type of "intelligent" systems developed for technology-enhanced learning. The first systems mainly used knowledge about the domain to support learning (i.e., the content was structured according to experts' knowledge). However, such systems have continually been criticized for believing that this is sufficient for effective learning to occur. In reality, these early systems constrained the learner because of fixed learning strategies and limited possibilities for the learner to investigate topics the ITS believed to be of no relevance. Later ITSs not only used knowledge about the domain but also about the learner, and about teaching strategies, in order to support more flexible individualized learning and tutoring (Brusilovsky, 1998). One of the goals of these ITSs was to adaptively deliver content. Most of these ITSs followed an integrated approach and merged the information about the domain, the teaching strategies, and the learner into one single model. Using, for example, a different learning strategy or underlying pedagogical model involved re-authoring of the complete model. This turned out to be very inflexible because the learning content was difficult to reuse and the engine was too domain-specific.

Constructivist learning environments support different learning strategies

In addition, ITS were said to be contradictory to the constructivist view of learning (Wasson, 1996). ITS focus on planning *precise* instructional steps by means of analyzing the learner's state of knowledge in terms of the learner's correct knowledge or misconceptions – the design of a constructivist learning environment focuses more on providing *different* learning strategies, e.g., by different learning paths. Therefore, Akhras and Self state that system intelligence should move to a more constructivist view of learning, i.e., moving from the product to the process of learning by shifting away from a model of "what" is learned towards a model of "how" knowledge is constructed (Akhras & Self, 2000).

The next generation of intelligent applications for learning were developed under the label of adaptive *hypermedia system* (AHS) or more specifically, *adaptive educational hypermedia system* (AEHS).

Adaptive hypermedia systems adapt the content to the user's knowledge and goals

"By adaptive hypermedia systems we mean all hypertext and hypermedia systems which reflect some features of the user in the user model and apply this model to adapt various visible aspects of the system to the user (Brusilovsky, 1996)." Adaptive hypermedia systems (AHS) have enhanced classical hypermedia by using an intelligent agent that supports a user during work with hypermedia. The intelligent agent is able to adapt the content of a hypermedia page to the user's knowledge and goals or suggest the most relevant links to follow (Brusilovsky, 2001). Compared to ITS, *Adaptive Hypermedia* is, in fact, a more recent research domain than ITS (Brusilovsky, 1996). AHS allow the adaptation of learning to specific user needs and requirements. They apply different types of learner models to adapt the learning content and the links of hypermedia pages to the user (Brusilovsky, 1998). Some well-known systems are AHA!2.0, ActiveMath, ELM-ART, INTERBOOK, and KBS Hyperbook (see Section 3.4.1.3).

In the following, the terms adaptivity and adaptation are defined and an introduction to adaptive concepts is given, which are then detailed in Section 3.4.1.2 by relating them to available methods and techniques.

3.4.1.1 Adaptivity and Adaptability

Technology-enhanced learning systems that adapt the learning experience to each individual are called *personalized e-learning systems*. They allow the learner to control his learning process to some extent and provide functionalities to personalize his learning experience. The personalization of learning may involve the tailoring of contents, tools, communications, connections, etc. to the needs of the learner. These systems can be categorized into two techniques: *adaptability* and *adaptivity (Oppermann, Rashev, & Kinshuk, 1997)*. Adaptability means that the personalization and all modifications are controlled and steered by the user. Adaptivity refers to an automatic personalization done by the system. Adaptivity is synthetic, a posteriori, whereas adaptability is analytic, a priori. The main advantage of adaptable systems over adaptive systems is that they give the users control over the process of adaptation and reduce the effect of incorrect system decisions. Hence, even if the main interest of the adaptive hypermedia research community lies on adaptivity, the challenge of adaptive hypermedia systems is to find a balance between adaptability and adaptivity, i.e., to find a mix between user-driven and system-driven personalization. Personalization of learning could be beneficial in terms of time, money, and effectiveness (O. Conlan, Dagger, & Wade, 2002).

Different paradigms of personalization

Dagger, Wade, and Conclan list different paradigms for personalization: *Context Personalization* is adapting to the preferences of the individual and to the learner's learning, respectively working, environment. *Competency personalization* is adapting to the prior knowledge of the learner in a specific domain. *Prerequisite personalization* is adapting to the currently required prerequisites of the learner, such as chosen learning objectives and learning goals (Dagger, Wade, & Conlan, 2003). Dagger mentioned that when information is adapted to a specific device, this context personalization is called *terminal adaptivity* (Dagger et al., 2003). Again, personalization may refer to adaptivity as well as adaptation. Brusilovsky names six classes of indicators to adapt to. They refer more to personal attributes and less to contextual or technical ones: a) background knowledge (e.g., language skills, experiences with the e-learning environment, etc.), b) domain-specific knowledge (e.g., knowledge about the content domain), c) cognitive and affective abilities (e.g., intellect, learning speed, motivation, etc.), d) constitutional attributes (e.g., physical properties of the body, concentration, age, etc.), e) preferences (e.g., preferred presentation or navigation mode, etc.), and f) interests and learning targets (e.g., user intention to use the system, learning goal, etc.) (Brusilovsky, 1996).

If there are changes to the indicators above, the system can provide methods and techniques to adapt the learning experience. The methods and techniques are presented in the next section.

3.4.1.2 Adaptation Methods and Techniques

Over the years, many different adaptation methods and techniques have been developed. Several classifications exist that are mostly based on the classification of Brusilovsky (Brusilovsky, 2001). In 1993, he distinguished between four forms of adaptation, namely adaptive navigation, structural adaptation, historical adaptation, and adaptive presentation (Brusilovsky, Pesin, & Zyryanov, 1993).

- *Adaptive navigation* intends to guide the learner through the system by changing the structure presented to the learner according to the individual learner needs and preferences.
- *Structural adaptation* aims to give the student a spatial representation of the hyperspace. This representation is based on the learner model and attempts to provide the student with a sense of position within the learning environment.
- *Historical adaptation* includes traces through the system, landmarks made during the learning process, and progression status generated by the system, which correspond to a specific learner.
- *Adaptive presentation* refers to adaptation of the way content is visually displayed to the learner based on the learner model.

Adaptive navigation and adaptive presentation

Later, in 2001, Brusilowsky differentiated between *adaptive course navigation support technologies* (i.e., link-level adaptation), which support the student regarding orientation in a hyperspace and navigation by changing the appearance of visible links (this subsumes the first three categories of classification from 1993), and *adaptive presentation technologies* (i.e., content-level adaptation), which adapt the content of a hypermedia page to the user's goals, knowledge, and other information stored in the learner model (Brusilovsky, 2001).

Methods and techniques for implementing adaptation and adaptability

In the following, methods and techniques for both forms of adaptation are elaborated. The methods describe the general concept of adaptation and the necessary sequence of interrelated steps of actions, each of which is carried out according to one or more techniques. The techniques implement the steps of a method. Brusilowsky defines a technique through a user model representation and an adaptation algorithm (Brusilovsky, 1996).

1. Methods and techniques for adaptive navigation

Adaptive navigation alters the navigation structure

By knowing the user's goals and knowledge, an adaptive hypermedia system can support users in their navigation by limiting browsing space, providing adaptive comments to visible links, or just suggesting the most relevant links to follow. Adaptive navigation alters the structure presented to the learner according to the individual learner characteristics. The most popular methods of adaptive navigation are (Brusilovsky, 2001):

- *Global guidance*: deals with identifying of the shortest way to reach a goal, e.g., a learning goal. A recommendation of links can be a solution for this method.
- *Local guidance*: deals with assistance in just one navigation step, e.g., the choice of the next, best learning chunk.
- *Global orientation*: The user should be able to understand the global hypertext structure and his current position better, e.g., through sitemaps or coloring of the pages already visited.
- *Local orientation*: improving the understanding of the local hypertext. The links that are not relevant for the current goal or because of the lack of experience of the user can be hidden with this method structure.
- *Personalized views*: deals with displaying an optimized view for the current user of the hyperspace. Only the relevant links are presented to the user.

Six adaptive navigation techniques are distinguished:

- *Direct guidance*: displays the best learning chunk for the learner to visit. This can be done by offering only one link to the best alterna-

tive. Direct guidance always follows a certain goal, e.g., a learning goal.

- *Sorting links*: deals with sorting links into the order of most relevance. This could be done according to the learning difficulty, or to learner preferences, learning goals, etc.
- *Hiding links*: hiding links of irrelevant pages. Learning chunks that have prerequisite concepts to be learned first by the learner may be hidden, or links to already known topics. Links can either be completely removed (*removing links*), disabled (*disabling links*), or hidden (*hiding links*).
- *Annotation links*: deals with augmenting links with personal dynamic comments in any form. This could be icons, textual tool tips, etc.
- *Link generation*: generating new links according to the context or user model that were not authored during the hyperspace development. For example, links to descriptions by experts can be generated automatically according to the learner model.
- *Map adaptation*: changes the navigation structure of the hyperspace. Maps usually graphically represent a hyperspace or a local area of a hyperspace as a network of nodes connected by arrows. These maps can be adapted to the learner needs. An example of such maps are hyperbolic trees.

The following table shows which techniques can be used for implementing the different adaptive navigation methods.

Table 4 Methods and techniques for adaptive navigation

Technique / Method	Direct guidance	Link sorting	Link hiding	Link annotation	Link generation	Map adaptation
Global guidance	+	+			+	
Local guidance	+	+	+	+	+	
Global orientation			+	+	+	+
Local orientation		+	+	+	+	+
Personalized views		+	+			+

It can be seen that especially the techniques related to links are suitable for implementing several methods. They are also the most commonly used techniques in adaptive hypermedia systems. Implementing map adaptation is much more complex.

2. Methods and techniques for adaptive presentation

Adaptive presentation alters the visualization of content

Adaptive presentation refers to content adaptation and alters the way content is visually displayed to the user based on different models. It deals with different forms of content explanations and sorting of information chunks. Different content explanations may refer to displaying additional information chunks, showing additional information based on

prerequisites, or showing related topics. The most popular methods of adaptive presentation (Brusilovsky, 2001) are:

- *Additional explanations:* insert the suitable low-level details into the fragment of information or leave them aside. E.g., examples could be optionally displayed in addition to more general explanations of a topic if the learner is able to understand them.
- *Explanation variants*: show different versions of the document, or a learning chunk. For example, according to the knowledge of the learner certain parts of a document (e.g., learning chunks) could be left out.
- *Sorting*: sorts the fragments of information and emphasizes the more important information chunks. For example, learning chunks could be shown according to their difficulty or length.
- *Prerequisite explanations*: explain additional concepts before displaying a specific fragment of information, if they are not known yet. Example: The general concept of software quality assurance is explained before the more specific topic of software inspections is presented.
- *Comparative explanations:* show only the differences and similarities of a new concept if a related concept was already shown before. Example: Show the difference and similarities of inspections and reviews.

Five adaptive presentation techniques are distinguished:

- *Stretchtext*: deals with expanding and collapsing paragraphs to reveal or hide details. Example: Text is hidden if it is not relevant for the current context or non-essential text, such as footnotes, is hidden and only shown when the learner selects it.
- *Conditional text*: The text is only displayed when certain conditions are fulfilled, for example, when specific pre-requisites are fulfilled.
- *Page variants*: deals with the presentation of different variants of a whole page. Example: The learning chunks on a page are sorted in different ways and each alternative can be presented.
- *Fragment variants*: similar to page variants. This deals with the adaptation of parts of a page, e.g., exchanging single learning chunks on a specific page.
- *Frame*-based *technique*: deals with the aggregation of several information/learning chunks to form a hypermedia document. A frame is a kind of container consisting of several slots. Each slot can be used to present an information chunk. The selection and sequence of the slots to be displayed depends on the user model or on the characteristics of the information chunks (e.g., length, difficulty).

Table 6 shows which techniques can be used for implementing the different adaptive presentation methods.

Table 5 Methods and techniques for adaptive presentation

Method \ Technique	Stretchtext	Conditional text	Page variants	Fragment variants	Frame-based technique
Additional explanations	+	+			+
Explanation variants		+	+	+	+
Sorting					+
Prerequisite explanations	+	+			+
Comparative explanations		+	+		+

With new web technologies, such as Ajax (i.e., Asynchronous JavaScript and XML), techniques such as stretchtext can be implemented more easily than a few years ago. The most powerful technique is the frame-based technique, because all methods can be realized.

3.4.1.3 Adaptive Educational Hypermedia Systems

Three main application areas of AHS

Brusilovsky names three application areas for AHS. The first application area is online documentation systems. The second area is application systems with advanced help and explanation facilities. The third application area is educational systems (Brusilovsky, 2001). AHS may tailor the educational offerings to the learner's objectives, prior knowledge, experience level (Pérez, Lopistéguy, Gutiérrez, & Usandizaga, 1995) (Hockemeyer, Held, & Albert, 1998; Milosavljevic, 1997), learning style (Gilbert & Han, 1999; Specht & Oppermann, 1998), and many other characteristics of the learner.

Examples of AHS

Some systems have been developed ten years ago and have been improved over many years, e.g., ELM-ART II (G. Weber & Specht, 1997). INTERBOOK is an approach for authoring and delivering adaptive electronic textbooks on the Web (Brusilovsky, Schwarz, & Eklund, 1998). ActiveMath (Melis et al., 2001) or AHA!2.0 adapt hypermedia pages based on conditional fragments (P. De Bra, Aerts, Smits , & Stash, 2002). KBS Hyperbook (N. Henze & Nejdl, 2001) is an adaptive hypermedia system that "guides students through the information space individually by showing the next reasonable learning steps, by selecting projects, generating and proposing reading sequences, annotating the educational state of information, and by selecting useful information, based on a user's actual goal and knowledge" (N. Henze & Nejdl, 2000). KBS Hyperbook follows a goal-driven approach that uses a Bayesian network technique for its user model. NetCoach (G. Weber, Kuhl, & Weibelzahl, 2001) is the follow-up to ELM-ART II (G. Weber & Specht, 1997). It provides a framework for building adaptive hypermedia systems. NetCoach uses a knowledge base consisting of concepts, which is the basis for adaptive navigations support. "These concepts are internal representations of pages that will be presented to the learner" (G. Weber, Kuhl et al., 2001). Most of these systems are rule-based systems, i.e., the adaptation is done by executing rules when certain conditions are fulfilled.

Reference models of AHS

There exist several reference models for adaptive hypermedia systems, e.g., the AHAM Reference Model (P. De Bra, Houben, & Wu, 1999) and the Munich Reference Model (Koch, 2000), which are both informal, respectively semi-formal, models (i.e., based on diagram techniques and natural language). Both the AHAM and the Munich Reference Models extend the Dexter Hypertext Model (Halasz & Schwartz, 1994), and describe a framework for describing the different components of adaptive hypermedia systems. The focus of these reference models is on process modeling and on the development of adaptive hypermedia applications, i.e., they provide process-oriented descriptions of adaptive educational hypermedia systems. The Dexter Hypertext Model was one of the first reference models written in Z, which provided a basis for many successors.

Typical architecture of AHS

The architecture of an adaptive hypermedia system (e.g., AHA!) usually consists of three components (P. De Bra, Stash, & De Lange, 2003): a *domain model*, which refers to the concepts with its relations and the learning goals; a *user model* which represents the user's characteristics, and an *adaptation model*, which consists of adaptation rules. These rules define how the concepts and related content resources are selected and adapted according to the learning process.

Furthermore, recent developments in the domain of the Semantic Web have led to some first approaches that introduce adaptive functionality to the Semantic Web. Example can be found in (Brusilovsky & Nijhawan, 2002), (Conlan, Lewis, Higel, O'Sullivan, & Wade, 2003), (Dolog, Henze, Nejdl, & Sintek, 2003), or (Frasincar & Houben, 2002). In addition, the Semantic Web and its related technologies such as RDF, RDFS, OWL, etc. offer up new possibilities for developing AHS. However, these systems suffer from several problems, which are elaborated in the next section.

3.4.1.4 Problems of Adaptive Hypermedia Systems

AHS should be able to facilitate the task of a user without having negative effects when adaptive actions are not precise. Moreover, an adaptation technique should provide some level of control, offer sufficient transparency, and reduce the risk of disrupting the user's understanding, mental model, and orientation in the hyperspace.

Three main problems of AHS

However, adaptive systems suffer from three general problems (Tsandilas & Schraefel, 2004) that can also be defined as the major challenges in developing AHS:

- They depend on the construction of user models that are incomplete and usually erroneous.
- They result in complex conceptual models that cannot be comprehended by users.

- They may prevent users from having control over the system's actions.

User models cannot accurately describe the user

Although research in the areas of user modeling and machine learning tries to address the first problem, it is commonly known that no user model can accurately describe a user. Furthermore, future intelligent systems will not be able to predict precisely what users want. The second problem refers to the way an adaptive system makes decisions and acts. This process might not be clear to user because user models are hidden from the user. Hence, the actions taken by the system may seem inconsistent and unpredictable. The third problem becomes critical when the system cannot accurately infer the user's goals and needs, which can change rapidly. The system will not have enough evidence to capture any shift in the user's goals, unless the user gives detailed feedback. Based on these three general problems, more specific problems can be stated:

R-Problem 4a: Current intelligent systems do not follow a constructivist view of learning

First, all Intelligent Tutoring Systems (ITS) and many of the previously mentioned AHS are contradictory to the constructivist view of learning (compare with (Wasson, 1996). For example, ITS focus on providing *precise* instructional steps by means of analyzing the learner's state of knowledge in terms of the learner's correct knowledge or misconceptions, whereas the design of constructivist learning environment focuses more on providing *a variety* of learning paths. Therefore, Akhras and Self state that system intelligence should move towards a more constructivist view of learning, i.e., moving from the product to the process of learning by shifting away from a model of "what" is learned towards a model of "how" knowledge is constructed (Akhras & A., 2000).

R-Problem 3b: The "click&go" metaphor does not support learning and understanding

Second, Schraefel et al. argue that systems based on a hypertext paradigm using the 'click&go' metaphor for navigating through an information space are rather weak in terms of Bloom's learning goal taxonomy (Bloom, Engelhart, Furst, Hill, & Krathwohl, 1956) because navigating from one fragment to the next based on some semantic relationship does not support understanding and learning (Schraefel, Carr, & De Roure, 2004). This is conformant to the second problem stated in Section 1.2. Nevertheless, focusing on learning processes does not undermine the need for knowledge representation and user modeling in AHS.

R-Problem 2a: Learning content and adaptive functionality are not reusable

Third, AHS have a common problem that limits the reusability of their learning resources. This limitation is due to the design of these systems: The learning resource is intertwined with the logic for generating adaptive learning experiences. High cohesion limits the reusability of that learning resource, as the embedded logic often has dependencies on other learning resources.

R-Problem 2b: Adaptive functionality is strongly related to a "closed content corpus"

In addition, the problem that adaptive functionality is not reusable is related to the so-called open corpus problem in adaptive (educational) hypermedia (Brusilovsky, 2001; N. Henze & Nejdl, 2000). So far, adaptive hypermedia systems have worked on a closed set of documents (closed corpus); the documents are fixed at the design stage of the system, and alternations or modifications are hard to process. This widely used closed corpus explains why the document space can carry all this adaptation-related information. On the other hand, this approach does not allow opening up the document space or even work in an open environment like the Web.

Problem of adaptation rules

Fourth, another problem, which is also related to the strong cohesion between system components, refers to the relationship between learning resources and user characteristics. These relationships are often too complex to model and cover them all. This complexity leads to a number of problems with adaptation rules in rule-based systems (Wu & De Bra, 2001) (Karampiperis & Sampson, 2005):

- inconsistency, if several rules are conflicting,
- confluence, if several rules are equivalent,
- insufficiency, if one or several necessary rules are not defined, and
- because of the faulty cooperation of the adaptive rules it can happen, that the adaptation engine does not terminate.

Decision models can solves the problems of rule-based systems

There are different approaches that try to avoid the problems mentioned above. Wu and De Bra (Wu & De Bra, 2001) suggest so-called sufficient conditions. These constraints help authors to write adaptation rules that guarantee termination and confluence. Karampiperis et al. suggest abandoning the adaptive rules and using so-called decision models (Karampiperis & Sampson, 2005). The proposed alternative method at first generates all possible learning paths that correspond to the learning goal, and then selects the best one. The selection relies on a decision model, which estimates the suitability of learning resources for the user. A so-called decision-making function is used for estimating the suitability.

3.4.2 E-Learning Standards and their Support for Adaptivity

In many publications about learning resources, terminology issues are discussed, because there is a lack of consensus in the field of e-learning (Self, 1992), especially concerning learning objects. Numerous initiatives like AICC (the Aviation Industry CBT Committee), ADL (Advanced Distributed Learning), IEEE LTSC (the Learning Technology Standards Committee of the IEEE), and IMS Global Learning Consortium have made efforts to establish standards. Several years ago, a number of initiatives agreed to cooperate on the field of standards.

Learning objects (LO)

IEEE LTSC has developed the Learning Object Metadata Standard (LOM). This standard specifies the syntax and semantics of learning object metadata, defined as the attributes required to fully/adequately describing a learning object. *Learning objects* are defined here as "any entity, digital or non-digital, which can be used, re-used or referenced during technology supported learning (IEEE Learning Technology Standard Comittee, 2002)".

SCORM reference model

A huge amount of specifications are being developed by the IMS consortium. Several of these specifications have been incorporated and in some cases been adapted by ADL for defining the SCORM reference model, which is relevant for this work. SCORM describes a harmonized set of guidelines, specifications, and standards based on the work of several distinct e-learning specifications and standards bodies. SCORM provides a comprehensive suite of e-learning capabilities that enable interoperability, accessibility, and reusability of Web-based learning content. These specifications have one aspect in common: by separating the content from the structure and layout, they enable the author to develop different variants of learning material very efficiently, while relying on the same set of learning objects. *SCORM Sequencing and Navigation* provides techniques for sequencing learning objects by means of *Learning Activity Trees*, and the *IMS Learning Design* specification allows expressing more sophisticated pedagogical concepts by means of a more extensive role concept. User profiling is addressed by the *IMS Learner Information Package* specification (LIP). It allows tracking the learning process as well as storing the characteristics of the learner. Besides, it is possible to assign and update the competencies of a learner according to the specification *IMS Reusable Definition of Competency or Educational Objective* (RDCEO). Finally, the IMS *Question & Test Interoperability Specification* (QTI) allows describing tests and questions. The specification enables the exchange of different types of questions.

Currently, these standards do not explicitly provide mechanisms for supporting adaptivity. For example, there is no catalog of metadata for adaptivity in education for use in LOM, SCORM, etc. The main reason is that adaptive educational hypermedia systems are "too different" to generalize for a metadata-driven description (N. Henze & Nejdl, 2004).

SCORM allows defining assets for different aggregation levels and formats, languages and language levels, operating systems and platforms, interactivity types, semantic densities, intended end user roles, typical age ranges, difficulties, relations, etc. In spite of all these attributes, knowledge domains are not considered at all within SCORM, so the system cannot cross-reference from a content object to one or more content objects from other knowledge domains. In fact, SCORM separates learning content from its Learning Management System (LMS), which allows the usage of learning content by different LMS.

R-Problem 5: SCORM does not separate learning content from sequencing and student modeling

However, it does not separate learning content from sequencing and student modeling. In SCORM, learner modeling is "hardwired" into learning objects, i.e., the sequence of learning objects is the same for all learners. As a result, the adaptivity of content is very limited, since it is defined according to a specific learning approach, student type, and a specific set of learning objectives. In addition, it immediately excludes adaptive use of external content (open corpus), e.g., from the Web.

Using CMI for adaptation leads to conceptual and technical problems

In contrast, the interoperability standards, which ensure that different components of technology-enhanced systems can work together, are very relevant to adaptive systems when LMS systems are to be integrated with the AHS. The interoperability standard CMI, which was originally introduced by AICC (Aviation Industry CBT Committee, 1998) and later adopted by IEEE LTSC and ADL, is now part of SCORM (SCORM, 2006). CMI proposes a very comprehensive mechanism for communication between an LMS and a learning object. A learning object can store and query information about learner performance related to multiple educational objectives in an LMS. This is similar to the overlay model in ITS, which supports adaptation. In addition, an available course authoring tool can relate advanced sequencing rules to a structured set of learning objects. Hence, simple adaptation actions can be performed on the sequence of the learning objects. Nevertheless, using CMI for learner modeling leads to conceptual as well as technical problems (O. Conlan et al., 2002).

TEL Standards fall behind the state-of-the-art of AHS

Despite many advanced features introduced in SCORM and other standards, their support of personalized, technology-enhanced learning falls behind the state-of-the-art level in the field of AHS. Recently, researchers have tried to combine several standards while extending them with additional information (Dolog, Gavriloaie, Nejdl, & Brase, 2003), which again led to very proprietary systems with components and learning content that are difficult to reuse.

3.4.3 Instructional Design and Learning Objects Types

Instructional design (Am.) or didactic design (EU) is a pragmatic discipline

Many definitions exist for instructional design. All of them are an expression of underlying philosophies and perspectives of what is relevant in the learning process. Instructional design and development, as it has evolved in the United States as a discipline since the late 1950s, uses a systems methodology that focuses on clearly defined needs, goals, and testable systems. The Applied Research Laboratory of Penn State University defines instructional design as:

"the systematic development of instructional specifications using learning and instructional theory to ensure the quality of instruction. It is the entire process of analysis of learning needs and goals and the development of a delivery system to meet those needs. It includes development

of instructional materials and activities; and tryout and evaluation of all instruction and learner activities".

Instructional design models are often internally consistent, i.e., they define goals and methods for achieving those goals. In Europe, instead of "instructional design," terms like "didactic design" are preferred. Instructional design deals with setting up spaces in which human learners can be taught directly and where they can guide their own learning processes. It prepares learning paths such that learners with different prerequisites, with different needs, in varying moods and under widely unforeseeable circumstances can find their way (M. Memmel, Ras, Jantke, & Yacci, 2006). Memmel et al. stated that the purpose of instructional design is to change a learner's knowledge, skill, or attitude. So what kind of learning theories and principles should be followed?

Knowledge must be individually constructed from different learning paths

According to constructivist learning theories, knowledge cannot be transmitted to learners, but must be individually constructed and socially co-constructed by learners (Jonassen, 1999). Learning systems should provide learners with a wide range of services to assist and facilitate knowledge construction, because learners may construct their own meaningful understanding to a learning theme from different paths rather than exposing them to a particular learning method.

Instructional design should address problem-based learning

Research in cognitive psychology has shown that students learn better when they are involved in solving problems. *Cognitive Apprenticeship (Collins, Brown, & Newman, 1989)*, *Goal Based Scenarios* (Schank, Berman, & Macpherson, 1999), and the *4C/ID Model* (Merriënboer, 1997) are just three of the instructional models that address problem-based learning. Merrill proposed the *First Principles of Instruction*: Learning is facilitated when previous experience is *activated*, when the lessons learned are *demonstrated* to the learners instead of just presenting them, when the learners are required to *apply* their knowledge or skill to solve a problem, and when the learners are motivated to *integrate* the new knowledge or skill into their daily work (Merrill, 2000).

Cognition and context are strongly related Therefore, situated learning principles should be considered

Situated learning approaches developed mainly at the end of the 1980s emphasize that a human's tasks always depend on the situation they are performed in, i.e., they are influenced by the characteristics and relationships of the *context* (J. S. Brown, Collins, & Duguid, 1989). Because of the relationship between cognition and context, knowledge and the cognitive activities meant to create, adapt, and restructure the knowledge cannot be seen as isolated psychological products – they all depend on the situation in which they take place. Learning involves interpreting individual situations in the world based on one's own subjective experience structures. Learners have an active role and derive most of the information from real situations by themselves. This information is later integrated into their own knowledge structures. In consequence, a learn-

ing environment must be learning- and transfer-conducive, focusing more on situated learning principles by gearing the learning process towards the current working task.

Instructional design should be stored separately from the learning object

Instructional design for learning objects could be handled in two obvious places: (1) embedded within a learning object or (2) as a separate object (e.g., using IMS Learning Design). The best option is to handle the pedagogical rules of instructional design *outside* the learning objects. If the objects themselves are tagged and structured chunks of content, adequate variants could then be selected and sequenced in a different layer, providing for the highest degree of flexibility in adaptation. Learning object oriented instructional design is a challenge that has to deal with several issues such as syntax and semantics, granularity, and reusability (M. Memmel, Ras, Jantke et al., 2006).

R-Problem 1a: There exists no learning object type taxonomy for SE

By using the IEEE definition as a baseline, any developer or researcher should define certain types or categories of learning objects according to his special needs and interests. Friesen states: ″In order for the positive potential of learning objects to be realized, they need to be labeled, described, investigated, and understood in ways that make the simplicity, compatibility, and advantages claimed for them readily apparent to teachers, trainers, and other practitioners. (Friesen, 2004)″. However in software engineering no such learning object type taxonomy exists.

In order to define a taxonomy of learning object types that can be used in the learning space approach a literature survey has been done. Based on this literature survey, abstractions have been derived from the terminology used and definitions found. The resulting taxonomy is displayed in Table 6.

Learning objecs are classified into learning objectives, learning content, and situational content

The three main classes are learning objects containing *learning objectives*, *learning content*, or *situational content*. Learning objectives describe the learning goal of a learning experience offered or of a part of it. Learning content refers to the material that is presented to the learner for constructing factual knowledge, conceptual knowledge, and for stimulating the construction of procedural knowledge. Situational content refers to descriptions of situations/event of daily life or work by means of descriptions of products, processes, individuals, groups, etc. that were involved in a particular situation. The class of learning content is further refined into *fundamental*, *auxiliary*, *orientation*, and *resources*. Fundamental learning objects explain the basics of a topic taught. Auxiliary objects provide more details on the information provided by the fundamental learning objects. Orientation objects support the learner in understanding his current position in a learning space and the whole hyperspace or a specific part of it. Resource objects either provide cross references to other domain topics or links to external resources such as books, repositories, experts, etc. The next refinement of the classes will

be explained further in Section 4.2.1.2. These are used to derive the concrete learning object types used in a learning space.

It can be seen in Table 6 that most of the technology-enhanced learning approaches and standards provide classifications for learning content rather than for situational content, which again confirms that the approaches do not sufficiently address context in their learning environments.

Table 6 Classification of learning object types

	Learning object types																			
	Learning objectives	Learning content										Situational content							
		Fundamental			Auxiliary					Orientation	Resources	Product	Process	Project	Individual	Group	Organization	Customer	Software tool
		Definition	Description	Knowledge	Evidence	Explanation	Illustration	Interactivity	Assessment										
(IEEE Learning Technology Standard Comittee, 2002)	Learning objective	Instructional content	Instructional content	Instructional content	Instructional content	Instructional content	Instructional content	Instructional content	Instructional content	Instructional content	Instructional content				Person		Organization		Software tool
(Bannan-Ritland, Dabbagh, Murphy, & Wiley, 2002)					Case study	How-to, Interactive, sequenced tutorial, step-action table, job aid	Example	Modeling, show me, try me, simulation, collaborative activity, integrated practice	Assessment or pre-test, mastery test, check-your-understanding		Resources						Tool		
(Doan, Bourda, & Dumitrascu, 2006)		Definition, principle	Process, algorithm, rule, property	Law, (law of Nature), theorem, axiom		resume, remark, conclusion	Example	Exercise, evaluation	Question		Reference								
(IMS Global Learning Consortium, 2001)			Lecture					Exercise, simulation, experiment,	Questionnaire, self-assessment, exam	Problem statement									
(Kärger, 2006)			Policy, procedure	Law of nature, theorem	Math. demonstration, proof	Conclusion, introduction, remark	Counter-example, example	Exercise, exploration, invitation, real- world problem											
(Koper, 2003)		Knowledge object	Knowledge object	Knowledge object	Knowledge object	Knowledge object	Knowledge object		Test object	History, overview		Knowledge object	Knowledge object	Knowledge object					

	Learning object types																		
	Learning objectives	Learning content								Orientation	Resources	Situational content							
		Fundamental			Auxiliary							Product	Process	Project	Individual	Group	Organization	Customer	Software tool
		Definition	Description	Knowledge	Evidence	Explanation	Illustration	Interactivity	Assessment										
(Meder, 2001)		Theorem, formula definition, term definition, principle	Rule, procedure, checklist, strategy, degree	Law, law comment	Conclusion, proof	Assumption (hypothesis, idea), reflection	Example, counter-example			History, scenario (hypothetical situation, virtual world), facts, summary, overview	Cross-reference, document reference, communication reference								
(Cisco Systems, 1999)		Definition, principle statement	Procedure table, decision table, combined table, staged table, block diagram, cycle charts			Introduction, analogy, guidelines	Example, non-example, illustration, process demonstration	Instructor note, practice item	Assessment item	Fact list									
(Rech, Ras, & Decker, 2007b)		Definition, principle	Procedure, rule, checklist, strategy	Theorem, law		Analogy, explanation, model	Example, counter-example	Practice item, learning		History, overview, summary, scenario, space map	Reference								
(Weller, Pegler, & Mason, 2003)		Instructional object	Instructional object	Instructional object	Instructional object	Instructional object	Instructional object	Collaborative activity, technical activity, individual activity, assignment		Narrative objects									

4 Fundamental Modeling Concepts

"Learning is the process whereby knowledge is created through the transformation of experience" (David A. Kolb, 1984)

Developing the learning space approach follows two main research objectives: development of conceptual models on the one hand, and development of a method, techniques, and a system for generating learning spaces on the other hand. This section focuses on the conceptual models. The models are independent of the learning scenario and learning method. Section 5 will describe the technique for generating context-aware learning spaces by using these models.

Structure of this section

In order to develop the approach, requirements related to the objectives are listed in the next section. Section 4.2 explains the overall conceptual model consisting of the following packages: The learning space model consists of an instructional design model and a learning resource model that refer to the didactical structure, respectively the content artifacts, of a learning space (described in detail in Section 4.2.1.1 and Section 4.2.1.2); a context model and a domain model, which are required for realizing the context-aware adaptation of the learning space artifacts (Section 4.2.2 and Section 4.2.3); and, finally, a variability model, which defines the variabilities in the two learning space models (Section 4.2.4).

4.1 Research Objective and Requirements

In Section 1, three research objectives have been stated. The first one was related to the development of conceptual models:

Objective 1

Objective 1: Formally define conceptual models for enriching experience packages with additional information.

- Develop an instructional design model for the specification of the didactical structure of a learning space in order to realize different types of learning scenarios and methods.
- Develop a learning resource model with different abstraction layers that implements the instructional design model and defines the different types of artifacts used in a learning space.
- Develop a domain model for the consistent annotation of resources for learning spaces and extend the reuse model accordingly.
- Develop a context model for the description of situations in software engineering that covers all context dimensions required for the static

and dynamic adaptation of learning spaces and extend the reuse model accordingly.

- Develop a variability model for the specification and resolution of variants in a learning space on a structural as well as on a content level, which supports traceability amongst the different abstraction levels and separates adaptation information and functionality from learning resources.

The following section provides an overview of all the models before they are elaborated in the subsequent sections.

4.2 Reference Model of the Learning Space Approach

In order to support the understanding of the overall reference model and to understand the rationale behind each submodel, three scenarios are provided: The first scenario illustrates how an experience is annotated (i.e., provided with metadata) by the knowledge engineer (see Section 5 for the role descriptions); the second scenario shows how learning resources are documented and annotated by means of the domain model; the last scenario explains when and how a learning space is generated.

Scenario 1: Annotating an experience package

A knowledge engineer decides to document an experience in the domain of refactoring and describes how code smells of the type *long method* can be removed from Java code. He uses a template to document the experience package and describes the problem, the solution, and the benefit of applying the solution. Afterwards, he uses the keyword browser to select these keywords: Java code:product; refactoring:process, remove long method:process; code smell long method:knowledge; programmer:role, and Integrated Development Environment:software tool. These keywords stem from a domain ontology. Then, he selects concepts that describe the context of the situation where he has made the experience. This is done by relating the experience package to a concrete description of a Java system for which this refactoring has been made, to the development process he had followed, the project he was working on, and his personal individual profile. He saves the experience package in the experience base.

Scenario 2: Annotating learning resources

A knowledge engineer is responsible for searching for valuable content chunks in the knowledge management system that can be used for learning. She finds good examples of code smells of the type long method and decides to store them in the system as a learning resource. She reworks the examples and starts the *learning element authoring tool,* where she first has to select a resource type, i.e., *example*, and second, select appropriate keywords from a domain model browser, i.e., Java code:product; code smell long method:knowledge. She stores the learning resource in the repository.

Scenario 3: Generating a learning space during experience package reuse

A software developer has to refactor for the first time the code of his Java system. He decides to look for suitable experience packages and finds an experience package that describes how *code smells* of the type *long method* can be removed by the refactoring technique *remove long method*. However, he is not knowledgeable about refactoring at all and lets the system generate a learning space for his situation and the selected experience package. The system first selects a template for *experiential learning* from the instructional design repository and performs queries to resolve the variabilities in the learning space in order to adapt it to the current context. The queries are forwarded to the domain model as well as to the context model of the system. The queries' results let the system resolve the decisions in the decision model and let the system choose appropriate techniques for adapting the learning space. Then, the learning space is presented by means of a set of linked hypermedia pages: First, basic concepts of refactoring are explained by presenting learning resources of different types (e.g., definition, explanation, etc.) about the domain concept instance code smell long method:knowledge, the domain concept instance remove long method:process, etc. Then, the experience package description is extended by referring to the context concepts that were selected by the expert during documentation, i.e., the details of the product, process, project, individual, and software tool context concepts are shown.

A package structure describes the conceptual models

Several conceptual models describe the entities and relationships of the learning space approach, and form the basis for the approach described later in Section 5. Figure 9 shows the different models and how they are related. The Unified Modeling Language (UML) has been used to illustrate the package structure of the overall conceptual model.

Predicate logic is used to define the models' concepts and their relationships

In the following, formal definitions are given for the entities and relationships of the packages, i.e., the conceptual models. Predicate logic is used to describe the models, which use non-logical symbols (i.e., predicates, functions, and constants) and logical symbols (i.e., an infinite set of variables, logical operators, quantifiers) (see also (Hamilton, 1978)). The interpretation of the formal language is done by using a *model* (D, I), where *D* is the domain and *I* is an interpretation of the elements of *D*. The elements of *D* are predicates (denoted by uppercase letters *ABC*, *P*, *Q*, *R*, …), functions (denoted by lowercase letter *f*, *g*, *h*, …), and constants (denoted by lowercase letters *a*, *b*, *c*, …). Functions are defined as predicates (i.e., a function such as $f(x_1,x_2,...)$ will similarly be replaced by a predicate $P(x_1,x_2,...,y)$ that is interpreted as "$y=f(x_1,x_2,...)$"). Practical examples are given that make use of a specific notation, i.e., instances are underlined, classes use a normal font, and relations are written in italic. Instances and their class are separated by ":".

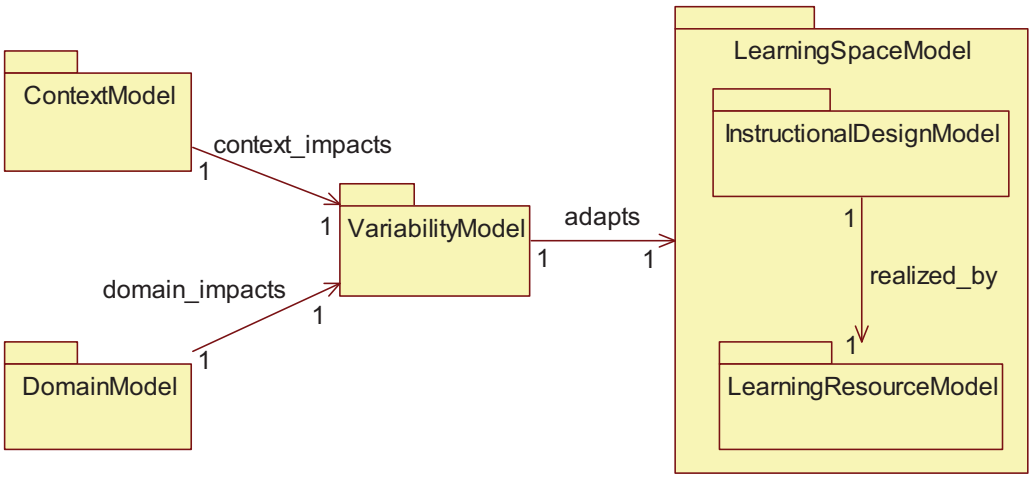


Figure 9 Overall conceptual model

A learning space realizes exactly one learning scenario

A learning space structures information in a didactical manner so that learning processes are supported. In the domain of learning theory, many different learning methods exist that can be specified and realized by means of a specific learning space model. A concrete learning space model implements exactly one concrete *learning scenario* lsc_i of the complete set of the *learning scenario domain* LSC on the instructional design and learning resource level. A concrete learning scenario lsc_i can be understood as the scope of a specific learning space.

Definition 3 Overall Conceptual Model

Overall Conceptual Model – The *OverallConceptualModel* OCM $\subseteq$ LSM x VM x CM x DM be the set of conceptual models for different learning scenarios LSC. That is:

$$\forall lsc\ OCM(lcs) \rightarrow \exists lsm\ LSM(lsm) \wedge \exists vm\ VM(vm) \wedge \exists cm\ CM(cm) \wedge \exists dm\ DM(dm)$$

with the predicates (i.e., relations):

$$CONTEXT\_IMPACTS(cm, vm) = \subseteq CM \times VM$$

$$DOMAIN\_IMPACTS(dm, vm) = \subseteq DM \times VM$$

$$ADAPTS\ (vm, lsm) = z \subseteq VM \times LSM$$

with CM being the set of context models, DM the set of domain models, VM the set of variability models, and LSM the set of learning space models. A learning space model consists of two submodels that define the instructional design of the learning space and the resources used within a learning space.

Definition 4 Learning Space Model

Learning Space Model – A *LearningSpaceModel* LSM consists of an *InstructionalDesignModel* idm and a *LearningResourceModel* lrm. Let LSM be the set of all potential learning space models for a specific learning scenario $lsc_i \in$ LSC with the relation:

REALIZED_BY (idm, lsc, lrm) = {idm | ∃(lsc ∈ LSC), ∀(lrm ∈ LRM) : idm realized_by lrm} ⊆ IDM x LRM

with IDM being the set of instructional design models and LRM the set of learning resource models. The learning resource model and the instructional design model are defined in detail in Section 4.2.1.1, respectively Section 4.2.1.2.

A learning space model is adapted by a variability model

As defined in Definition 3, the learning space model is adapted by means of a variability model. The latter is impacted by a context model and a domain model, where the information of those two models is used for resolving the variabilities in the variability model.

In order to support adaptation, extensions to the reuse model are necessary. Figure 10 shows the extensions by an arrow.

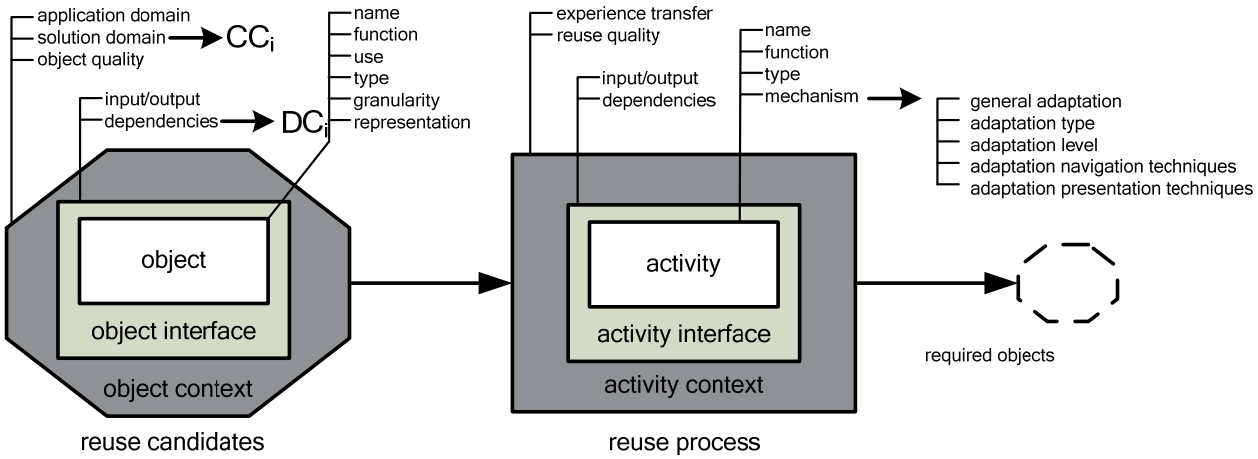


Figure 10 Reuse model of Basili and Rombach (V. R. Basili & Rombach, 1991)

Reuse candidates

The extensions to the reuse candidates are:

- *Object Interface* (Dependencies): Instead of an informal description of the dependencies, instances of domain concepts DC_i from a software engineering domain ontology reflect the most important facts used in the experience package.
- *Object Context* (Solution Domain): a more precise description by selecting context instances including their semantic relationships describes the context CC_i (e.g., instances of classes: product, process, project, individual etc.) where the experience package has been derived/documented.

The domain and context concepts and their semantic relationships will be defined more formally below.

Reuse process

The standard reuse process consists of the activities *identifying* reuse candidates from the reuse repository, *evaluating* the reuse candidates and *selecting* a candidate, *modifying* the candidate before reuse, if necessary, and finally *integrating* or applying the experience (V. R. Basili & Rombach, 1991). By following a learning space approach, a new activity of *generating* a learning space is inserted between the selection and the evaluation process.

In order to further specify the adaptation mechanisms, further sub-characteristics need to be added to *Mechanism*:

- *General Adaptation*: describes whether the adaptation is done by using a decision model or a feature model
- *Adaptation Type*: describes whether the adaptation is done only before runtime (i.e., before the learning space is presented to the user: static) or also during run-time (i.e., when the context is changing during the usage of the learning space: dynamic)
- *Adaptation Level*: describes on which level the adaptation takes place (i.e., structure, content, presentation)
- *Adaptation Navigation Techniques*: lists the different adaptation techniques which are used to perform the adaptations on the level of structure and hence which adapt the navigation structure within a learning space (see Section 4.2.4 for examples)
- *Adaptation Presentation Techniques*: lists the different adaptation techniques which are used to perform the adaptations on the level of content and presentation (see Section 4.2.4 for examples)

Definition 5 Context Model

Context Model – A *ContextModel* CM is used to describe a concrete software engineering situation that is determined by a finite set of context concepts and relations amongst instances of these context concepts $CM \subseteq CC_1 \times CC_2 \times \ldots \times CC_n$, where $CC_i \in CC$. CC is the complete set of possible context concept sets and with a relation:

$$P^n(relc_1, relc_2, \ldots relc_n) = \{(relc_1, relc_2, \ldots, relc_n) \mid relc_1 \in RELC_1, relc_2 \in RELC_2, \ldots, relc_n \in RELC_n\} \subseteq RELC_1 \times RELC_2 \times \ldots \times RELC_n$$

with $RELC_1, RELC_2, \ldots, RELC_n$ being the sets of relations between two context concepts:

$$RELC_k(cc_i, cc_y) = \{cc_i \mid \forall(cc_j \in CC_j) : cc_i \text{ relates\_to } cc_j\} \subseteq CC_i \times CC_J$$

with $CC_i \times CC_J$ being two disjoint sets of context concepts.

Example: Context Model

Hence, a situation can be described using concrete instances of one to several context concepts and relations (e.g., Eric Ras:Individual *works_in* Open Source Practica 2007:Project and *produces* Digital Care Giver Assistant (DCGA) 1.0:Product).

Definition 6 Domain Model

Domain Model – A *DomainModel* DM is used to define the domain classes of the software engineering domain (i.e., domain concepts) and to define relations amongst instances of these sets of domain concepts. Therefore $DM \subseteq DC_1 \times DC_2 \times \ldots \times DC_n$, where $DC_i \in DC$. DC is the complete set of possible domain concepts.

$$Q^n(reld_1, reld_2, \ldots reld_n) = \{(reld_1, reld_2, \ldots, reld_n) \mid reld_1 \in RELD_1, reld_2 \in RELD_2, \ldots, reld_n \in RELD_n\} \subseteq RELD_1 \times RELD_2 \times \ldots \times RELD_n$$

with $RELD_1$, $RELD_2$,…, $RELD_n$ being the sets of relations between two domain concepts:

$$RELD_k(dc_i, dcj) = \{dc_i \mid \forall(dc_j \in DC_J) : dc_i \text{ relates\_to } dc_j\} \subseteq DC_i \times DC_J$$

with $DC_i \times DC_J$ being two disjoint sets of domain concepts.

Example: Domain model

That is, the domain model has several main classes that each reflect a particular concept type in software engineering. Relations between the instances of these concepts are used to formalize the body of knowledge in software engineering (e.g., the programmer:role *produces* java code:product and *needs_knowledge_ about_knowledge* code smell comment:knowledge).

The variability model plays a central role in the overall conceptual model. It specifies the variabilities of the entities and relations in the learning space models and describes the queries to the context and domain model in order to resolve the variation points (see Section 4.2.4).

Definition 7 Variability Model

Variability Model – A *VariabilityModel* VM defines the variabilities of the LearningSpaceModel, i.e., the non-common characteristics of a set of InstructionalDesignModels IDM and the non-common characteristics of a set of LearningResourceModels LRM for a concrete learning scenario ($lsc \in LSC$), which can be seen as the scope of the variabilities.

Definition 8 Variability

Variabilities for instructional design models idm_i i =0..n, where n is the total number of available instructional design models, are defined as:

$$VARIABILITY(idm, lsc) = \cup LSC(idm_i, lsc) - COMMONALITY(idm_i, lsc)$$

where the predicate LSC reflects all instructional design model variants idm_i that satisfy the learning scenario lsc:

$$LSC(idm, lsc) = \{lsc \mid \forall(idm \in IDM) : idm \text{ satisfies } lsc\}$$

and for learning resource models as:

$$VARIABILITY(lrm, lsc) = \cup LSC(lrm_i, lsc) - COMMONALITY(lrm_i, lsc) \text{ and}$$

$$LSC(lrm, lsc) = \{lsc \mid \forall(lrm \in LRM) : lrm \text{ satisfies } lsc\}$$

Definition 9 ommonality

Commonalities for instructional design models are defined as:

$$COMMONALITY(idm, lsc) = \cap\ CONTEXT(idm, cc_k, dc_l)$$

where the predicate CONTEXT means that a concrete idm satisfies all objects (k=1..m; m=number of all instances of a context concept set) of all defined context concept sets CC_i of a context model and satisfies all the objects (l=1..s; s=number of all instances of a domain concept set) of all defined domain concept sets DC_i of the domain model in the scope of a specific learning scenario lsc.

$$CONTEXT(idm, cc, dc) = \{idm \mid \forall(cc \in CCi)\ \forall(dc \in DCJ) : (idm\ satisfies\ cc) \wedge (idm\ satisfies\ dc)\}$$

and for learning resource models as:

$$COMMONALITY(lrm, lsc) = \cap\ CONTEXT(lrm, cc, dc)\ where$$

$$CONTEXT(lrm, cc, dc) = \{lrm \mid \forall(cc \in CC_i)\ \forall(dc \in DC_j) : (lrm\ satisfies\ cc) \wedge (lrm\ satisfies\ dc)\}$$

The variability model transforms IDM and LRM

Furthermore, the variability model transforms the instructional design model idm and the learning resource model lrm into idm', respectively lrm', based on information of a specific context model cm and a specific domain model dm (i.e., called indicators). The transformation function results in the following predicate that describes the transformation function:

$$VM(vm, cc, dc, idm, lrm, idm', lrm')) = \{idm, lrm) \mid \forall(vm \in VM), \forall(cc \in CC), \forall(dc \in DC) : f:(IDM \rightarrow IDM')\ V\ (LRM \rightarrow LRM')\}$$

with idm ∈ IDM and lrm ∈ LRM being two models that are transformed into idm' ∈ IDM' and lrm' ∈ LRM'. The subsequent subsections refine these five models and formally define their entities and relations. The following table shows an example of such a transformation on the structure level for a learning goal structure template (i.e., idm).

Table 7 Example of transformation on structure level (idm)

Impact Indicator (to what is it adapted?)	Value	Type	Generic Artifact to be adapted	Possible Artifact Variants (transformation)	Adaptation Level
Context experience package (CC_i)	• User is author • User worked in same project • User worked in same group	static	Learning goal structure template	• Only show remember level • Remember first situation and the context • Remember first situation and the context & ask colleague	• Structure •

4.2.1 Learning Space Model

A learning space is a hypermedia space with linked pages

From a technical point of view, a learning space consists of a hypermedia space with linked pages. A learning space follows a specific global learning goal and is created based on context information (and the context description of an experience package in the case of the experiential learning scenario). The learning space is technically presented in the Software Organization Platform (SOP: see also Section 6 and (S. Weber et al., 2008)).

As shown in the previous section, the package *LearningSpaceModel* consists of two subpackages: The subpackage *InstructionalDesignModel* 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.). The subpackage *LearningResourceModel* is dedicated to the specification of the content structures of a learning space. This model implements the concepts of the *InstructionalDesignModel* by filling templates with content and by creating physical links between instances of the conceptual artifacts (see Figure 11).

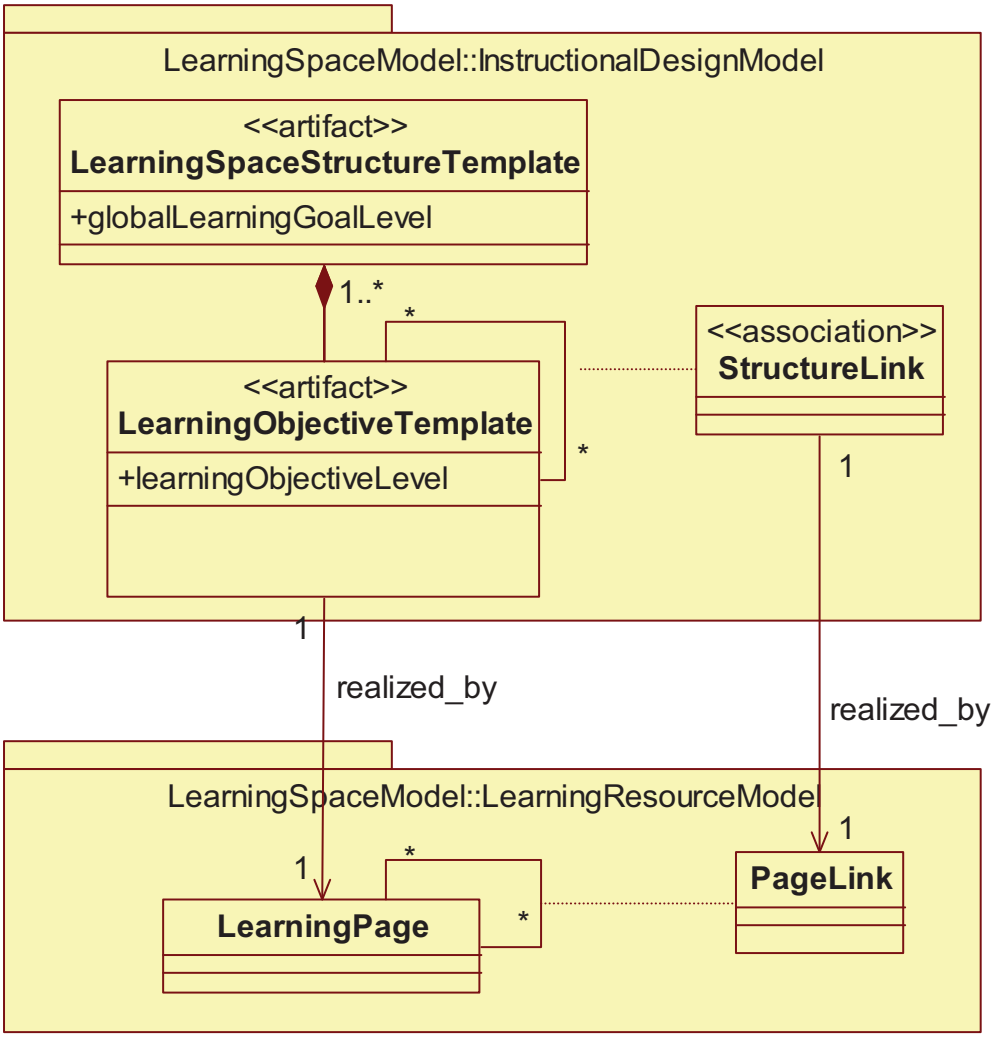


Figure 11 Learning space model (full realization dependency depicted in Figure 16)

4.2.1.1 Instructional Design Model

What an instructional design model is about

The instructional design model can be seen as the design of a learning space (see Section 3.4.3). It follows several instructional design strategies (see also Section 5.4.3) and implements one or several learning methods by 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; and c) providing a means for describing these templates, which are sequences of so-called learning activities.

Definition of learning method

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 practicum or workshop, learning by teaching, discussing with expert).

Learning goal vs. learning objective

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.

Figure 14 on page 87 shows a simple example of a learning space structure template.

Example: Instructional design model

> A global learning goal determines a concrete learning space structure template (e.g., remember refactor code smell comment:experience package). This template is refined by six learning objectives templates in this example. Each relates to a specific learning objective (e.g., LOT1: remember code smell comment:knowledge, LOT2: remember 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 context model and the domain model (see Section 4.2.2 and Section 4.2.3).

Levels of learning objectives

The approach uses Anderson and Krathwohl's taxonomy of educational objectives (L. W. Anderson & Krathwohl, 2001), which is a revision of the original taxonomy by Bloom (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 for 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 un-

derstanding. 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*.
- *Evaluating* also promotes transfer and means making judgments based on criteria and/or standards. The criteria used are mostly quality, effectiveness, efficiency, and consistency. The associated cognitive processes are *checking* and *critiquing*.
- *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 cognitive processes are *generating*, *planning*, and *producing*.

Learning space approach focuses on "remember, understand, and apply"

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.

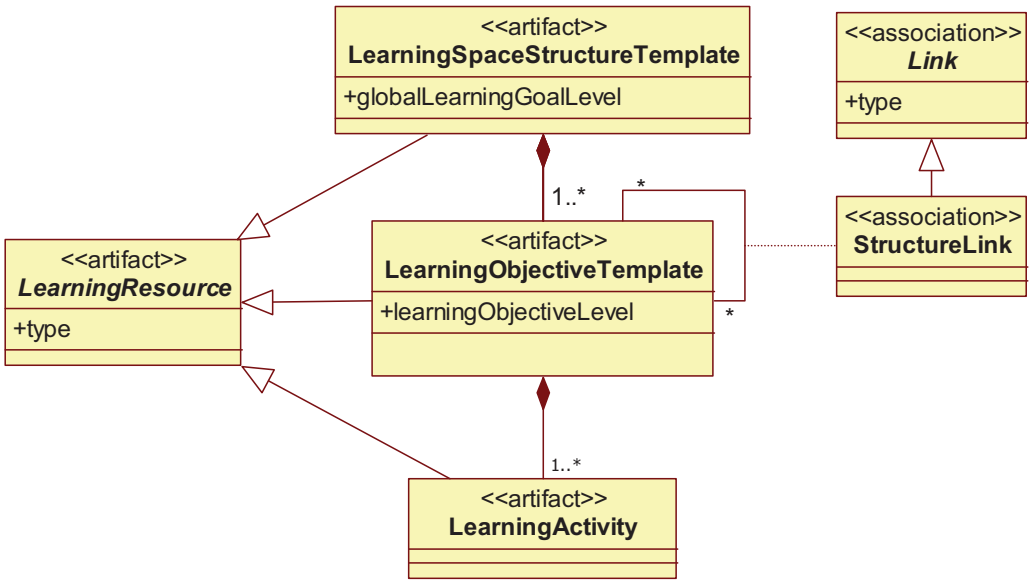


Figure 12 Instructional design model

The learning objectives are related by means of so-called structured links (see Figure 12), which are transformed into page links (i.e., physical hypertext links, see Figure 11). Then, 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 of how a global learning goal is refined into learning objectives and activities depends on the learning method to be followed.

Example: Learning activity tree

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 Figure 13 (remark: no specific learning method is implemented in this example and the activities are only related to the type "read" in order to keep the example as simple as possible).

Formal definitions are now provided for the conceptual artifacts and relations introduced.

Definition 10 Instructional Design Model

Instructional Design Model – An *InstructionalDesignModel* IDM specifies the structure of a learning space on a didactical level for a specific learning scenario. A learning scenario lsc reflects a learning method lm and is realized by a learning space structure template:

$$IDM(lsc, lm, lsst) = \{lsc \mid \exists(lsst \in LSST), \exists(lm \in LM) : (LSC(lsc, idm) \rightarrow LM(lm)) \wedge (lsst \text{ realizes } lsc)\}$$

with LSST being the set of learning space structure templates. For the predicate LSC, see Definition 7.

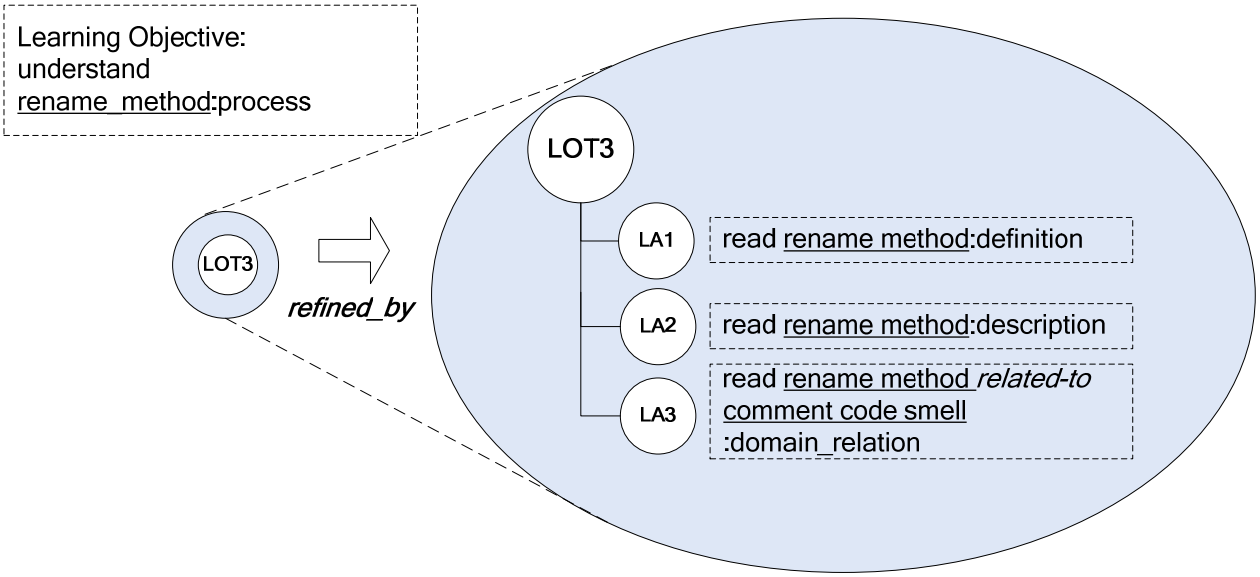


Figure 13 Example of learning objectives template with learning activities

Definition 11 Learning Space Structure Template

Learning Space Structure Template – A *LearningSpaceStructureTemplate* specifies the learning space on the highest level of abstraction by means of a global learning goal glg and a set of learning objectives templates lot ≠ empty set. The learning objectives templates can be linked by a structure link sl:

$$\forall lsst\ LSST(lsst) \rightarrow \exists\ lot\ LOT(lsst, lot)$$

Definition 12 Learning Objective Template

with LSST being the set of learning space structure templates and LOT the set of learning objective templates. This means that for each learning space structure template lsst, a learning objective template lot exists.

$$LOT(lsst, lot) = \{lsst \mid \forall lot \in LOT : lot\ refines\ lsst\}$$

Definition 13 tructure Link

Learning objective templates are linked by structure links sl:

$$SL\ (lot, sl) = \{sl \mid (\exists\ lot_1 \in LOT), (\exists\ lot_2 \in LOT) : lot_1\ structure\_link\ lot_2\} \subseteq LOT \times LOT$$

Each learning objective template lot consists of a finite set of learning activities la:

$$\forall lot\ LOT(lot) \rightarrow \exists\ la\ LA(lot, la)$$

Definition 14 Learning Activity

with LA being the set of learning activities. This means that for each learning objective template lot, a learning activity la exists:

$$LA(lot, la) = \{lot \mid \forall la \in LA : la \text{ refines } lot\}$$

4.2.1.2 Learning Resource Model

The learning resource model can be understood as the realization of the instructional design model. The conceptual elements of the instructional design model are further refined and instantiated by concrete content elements.

Learning resource model and learning space

Figure 14 shows an example of an instantiation of a learning resource model, which is called learning space. A learning space follows a specific global learning goal and is created based on context information about the current situation. The goal of a learning space is to provide a learning environment for self-directed situated learning at the workplace and follows a specific learning method that is specified in the instructional design model. The example of the instructional design model of the previous section has been used again in Figure 14 to show 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.

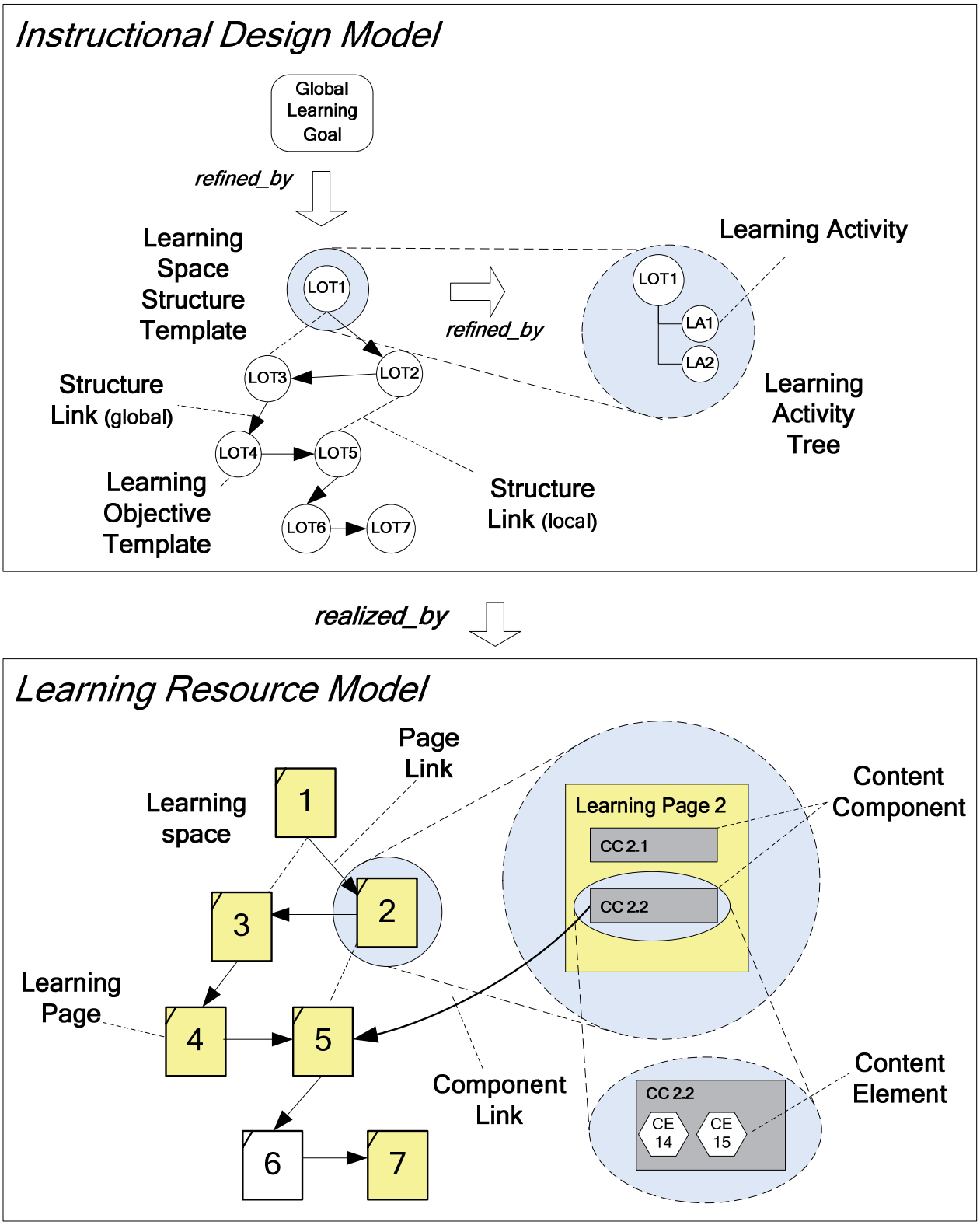


Figure 14 Example of learning resource model

Figure 15 presents an example of a learning page with the learning objective *remember* HideMethod:process with four instructional content components: references to other concepts are shown in the first content component, followed by an introduction, description, and example.

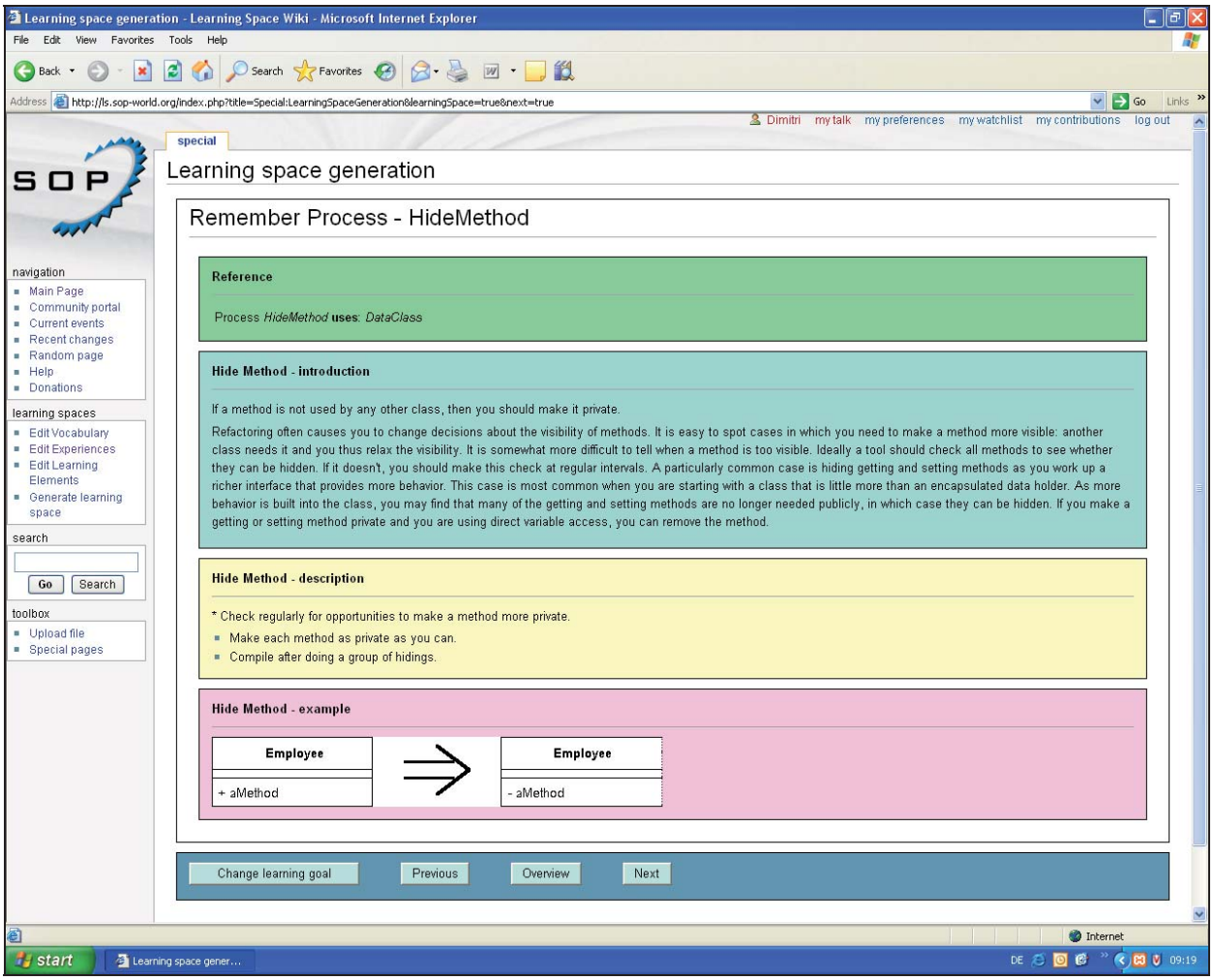


Figure 15 Learning page

Structure of a learning page

The package of the LearningResourceModel is depicted in Figure 16. A learning page is composed of so-called content components, which are composed of so-called content elements. A LearningPage can reference another LearningPage by using a PageLink. A ContentComponent can link another ContentComponent of other learning pages by using a ComponentLink. A ContentElement has no relations.

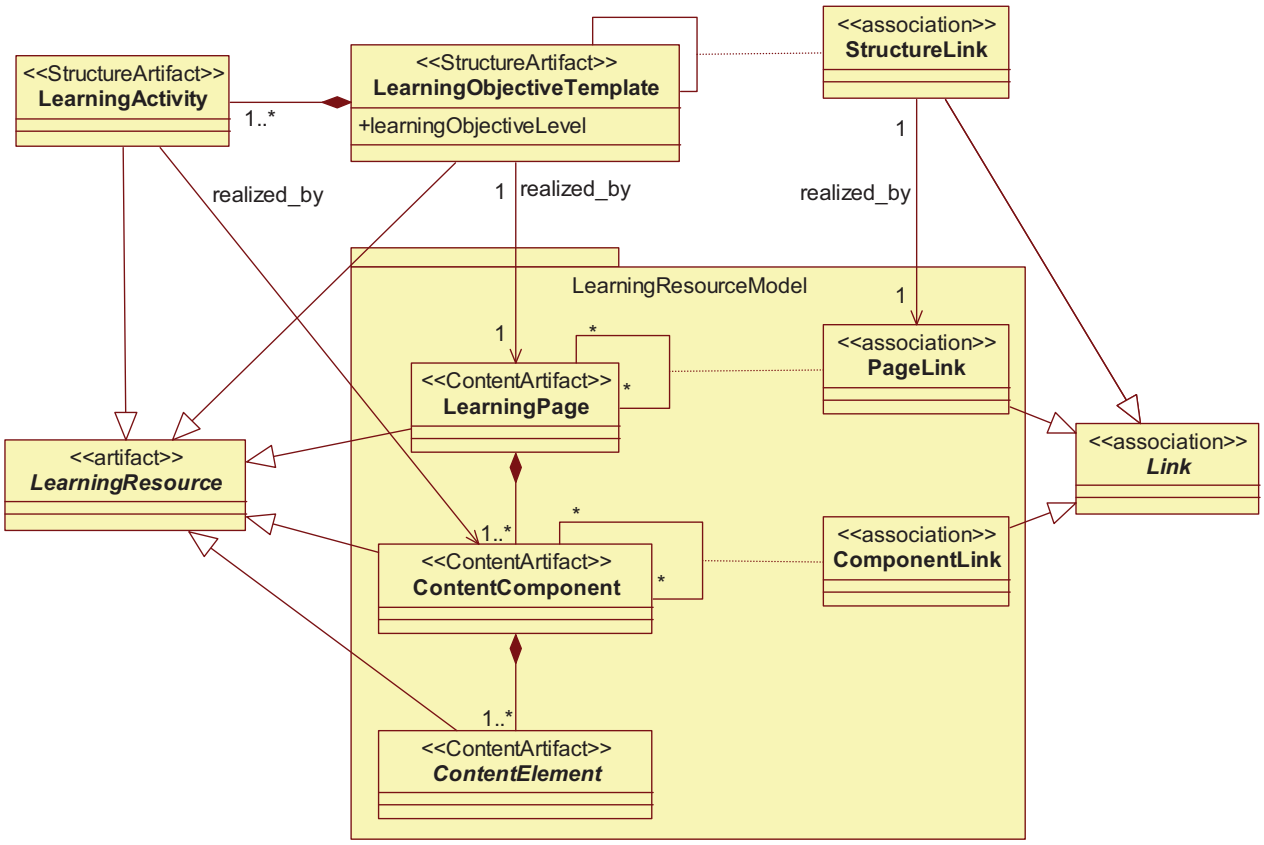


Figure 16 Learning resource model

Content components and 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 activity (see Figure 16). In addition, they can be referenced by another external content component (i.e., a content component of another learning page). The conceptual element ContentElement in Figure 16 is an abstract class. The inheritance structure is depicted in Figure 17.

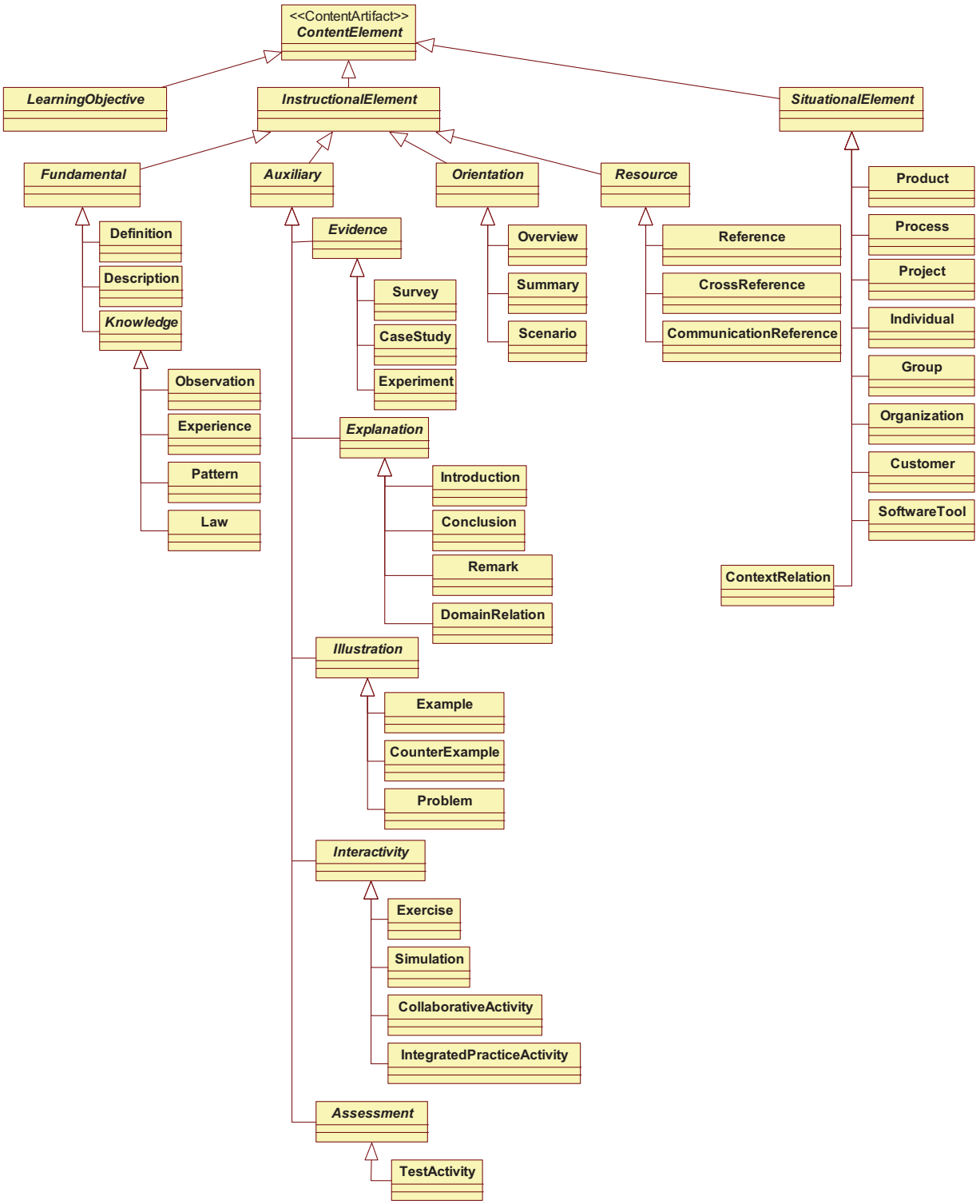


Figure 17 Content element types

The reason why the type of a content element type is not simply realized by a class attribute is that content elements of different types may have different content structures, different metadata sets, and different presentation modes.

By means of what has this type hierarchy been developed?

This type hierarchy has been derived from the literature survey on the classification of systems according to their learning object types in Section 3.4.3. In addition, special software engineering related types have been added, such as Evidence and its subclasses Survey, CaseStudy, and Experiment. In addition to the literature study, this type hierarchy has been extended from classifications of knowledge creating approaches in the field of software engineering, such as the experience factory (V. R. Basili, Caldiera, & Rombach, 2002) and other research work (Endres & Rombach, 2003), and from studying the description of comprehensive lifecycle and product models, such as the *V-Model XT* (Rausch, Broy, Bergner, Höhn, & Höppner, 2007).

The top level consists of LearningObjective, InstructionalElement, and SituationalElement. The LearningObjective element defines a learning objective of a LearningPage.

Instructional content elements

An InstructionalElement specifies explicit learning content and distinguishes 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 relation 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 ComponentLink). The following table shows two examples of instructional content elements.

Table 8 Examples of instructional content elements

Domain Concept	Domain Concept Instance	Type	Content
Process	Introduce-Assertion	Introduction	Often, sections of code work only if certain conditions are true. This may be as simple as a square root calculation working only on a positive input value. With an object, it may be assumed that at least one of a group of fields has a value in it. Such assumptions often are not stated but can only be decoded by looking through an algorithm. Sometimes the assumptions are stated with a comment. A better technique is to make the assumptions explicit by writing an assertion. An assertion is a conditional statement that is assumed to be always true. Failure of an assertion indicates a programmer error. As such, assertion

Domain Concept	Domain Concept Instance	Type	Content
			failures should always result in unchecked exceptions. Assertions should never be used by other parts of the system. Indeed, assertions are usually removed for production code. It is therefore important to signal that something is an assertion. Assertions act as communication and debugging aids. In communication, they help the reader to understand the assumption the code is making. In debugging, assertions can help catch bugs closer to their origin. It has been noticed that debugging help is less important when self-testing code is written, but the value of assertions is still appreciated in communication.
Knowledge	CodeSmell-Comment	Description	Comments should be used to give overviews of code and provide additional information that is not readily available in the code itself. Comments should contain only information that is relevant to reading and understanding the program and should be added when the author realizes that something is not as clear as it could be and adds a comment. Discussion of non-trivial or non-obvious design decisions is appropriate, but avoid duplicating information that is present in (and clear from) the code. It is too easy for redundant comments to get out of date. In general, avoid any comments that are likely to get out of date as the code evolves. In addition, the frequency of comments sometimes reflects poor quality of code. When you feel compelled to add a comment, consider rewriting the code to make it clearer. Some comments are particularly helpful: - Those that tell why something was done in a particular way (or why it was not) - Those that cite algorithms that are not obvious (where a simpler algorithm will not do). Other comments can be reflected just as well in the code itself! The refactorings ExtractMethod, IntroduceAssertion, RenameMethod should be used to remove this kind of code smells.

Situational content elements

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 have been created in a KM/EM system. Situational elements possess types that exactly correspond to the contextual concepts of the context model. In order to fulfill the requirements of integrating technology-enhanced learning and experience respectively knowledge management, on the informational level, situational content elements are merged with instructional content elements.

The learning resource model can be compared with the SCORM Content Aggregation Model (see Section 3.4.2). In the following, the conceptual artifacts of the learning resource model are defined formally.

Definition 15 Learning Resource Model

Learning Resource Model – A *LearningResourceModel* LRM realizes the structure defined by an instructional design model (see Definition 4). A learning resource model is defined as:

$$LRM(idm,lrm) = \{lrm \mid \forall(idm \in IDM) : lrm \text{ realizes } idm)\}$$

with IDM being the set of instructional design models. A learning resource model consists of a set of learning pages lp_i, which are composed of a set of content components cc_i. Content components are composed of a finite set of content elements ce_i that represent the physical content. The learning pages are linked by page links and content components are linked by component links. This results in the following predicates (i.e., relations):

$LP\_REALIZE\_LOT(lp, lot) = \{lot \mid \forall(lp \in LOT) : lot \text{ is\_realized\_by } lp\} \subseteq LOT \times LP$ (see also Definition 12 of learning objective template)

$CL\_REALIZE\_SL(cl, sl) = \{sl \mid \forall(cl \in CL) : sl \text{ is\_realized\_by } cl\} \subseteq SL \times CL$ (see also Definition 13 of structure link)

$CC\_REALIZE\_LA(cc, la) = \{la \mid \forall(cc \in CC) : la \text{ is\_realized\_by } cc\} \subseteq LA \times CC$ (see also Definition 14 of learning activity)

with LP being the set of learning pages, CL the set of component links, and CC the set of content components.

Definition 16 Learning Space

Learning Space – A *LearningSpace* physically represents the content presented to the learner, including its navigation structure. Hence, a learning space consists of different types of learning resources that are connected by different types of links:

$LS(ls_i) = \cap\, LR(lr_i) \times \cap\, LI(li_i)$ where

$lsi \in LS$, $lii \in LI$ with LS set of learning resources and LI set of links

Definition 17 Learning Resource

Learning resources are defined by:

$$LR(lr_i) = \cap\, LP(lp_i) \times \cap\, CC(cc_i) \times \cap\, CE(ce_i) \text{ and}$$

Definition 18 Link

Links are defined by:

$$LI(li_i) = \cap\, PL(pl_i) \times \cap\, CL(cl_i)$$

Definition 19 Learning Page

The predicate for a learning page LP can be defined as:

$$\forall lp\ LP(lp) \rightarrow \exists\ cc\ CC(lp, cc)$$

with LP being the set of learning pages and CC the set of content components.

Definition 20 Page Link

Page links are used to link learning pages:

$$PL\ (lp, lp) = \{lp \mid (\exists\ lp_1 \in LP), (\exists\ lp_2 \in LP) : lp_1\ page\_link\ lp_2\ \} \subseteq LP \times LP$$

with PL being the set of page links.

Definition 21 Content Component

For every learning page a content component exists, which can be linked by component links cl.

$$CC(lp, cc) = \{lp \mid \forall cc \in CC : lp \text{ is composed of } cc\} \text{ and}$$

$$CL\ (cc, cl) = \{cl \mid (\exists\ cc_1 \in CC), (\exists\ cc_2 \in CC) : cc_1\ component\_link\ cc_2\ \} \subseteq CC \times CC$$

with CC being the set of content components.

Definition 22 Content Element

For every content component cc, a content element ce exists.

$$\forall cc\ CC(lp, cc) \rightarrow \exists\ ce\ CE(cc, ce)$$

with CE being the set of content elements.

$$CE(cc, ce) = \{cc \mid \forall ce \in CE : cc \text{ is composed of } ce\}$$

In the following, the context model is defined, which also explains why specific types of situational content elements have been chosen in this section.

4.2.2 Context Model

Context is, in fact, a semantic element: A specific selection of information artifacts and their relations describe a particular situation or context. The context model is one of the two models that is used to resolve variabilities in the instructional design model and the learning resource model (see Section 4.2.4 for the variability model).

Context concepts need to be related to other instances to provide meaning

In the learning space approach, context artifacts are individual entities that exist in the software engineer's environment, e.g., processes, products, individuals, customers, etc. However, context artifacts really make no sense on their own. They inherently describe a real or virtual entity. For example, a product may be related to a process, a project, etc. A product on its own does not really have any value, unless we know to what other context artifacts it is related in order to build a comprehensive description of a situation.

Therefore, developing a context model is always a matter of choosing the context dimensions (i.e., context concepts) and a matter of defining relations between those concepts to adequately describe situations in software engineering.

The context model developed in this thesis has been developed during several projects at Fraunhofer IESE in the domain of reuse in software engineering. The model has been improved over the last years and the context model that is described next has been used for the learning space approach. In the *RISE project* (RISE stands for Reuse In Software Engineering) the *Riki* system consists of the six general classes of information: projects, products, processes, people, customers, and (further) knowledge (Rech, Ras, & Decker, 2006b; Rech et al., 2007b).

Reuse model extension

The context model extends the *solution domain* characteristic of the reuse model (see also Figure 10 in Section 4.2).

Dimensions of the context model

An earlier version of the model did not cover the software tool dimension (Ras, Rech, & Decker, 2006). The section about the state of the art of context dimensions (see Section 3.2.2), and the literature survey about the classification of KM/EM systems regarding the knowledge types they support (see Section 3.2.3) help to identify the context dimensions of the context model. Several approaches were used to reduce the selection of context dimensions to a "most useful selection" (Lenat, 1998; Mitchell, 2002; Pascoe, 2001; Schmidt, 2002):

- individual context (e.g., role, skill and competence profiles, learning preferences, activity history)
- group context (e.g., team size, team members, team competencies and experience)
- project context (e.g., size, effort, resources, costs, customer, contract, business domain)
- process context (e.g., activity, lifecycle model)
- product context (e.g., type of product, complexity, quality, application context)
- organization context (e.g., competence development strategies, corporate quality strategy, business targets)
- customer context (e.g., business domain, customer's products, turnover, etc.)
- software tool context (e.g., tools used for development, release and version, etc.)

A formal illustration of the context dimensions and their relations is given in Figure 18.

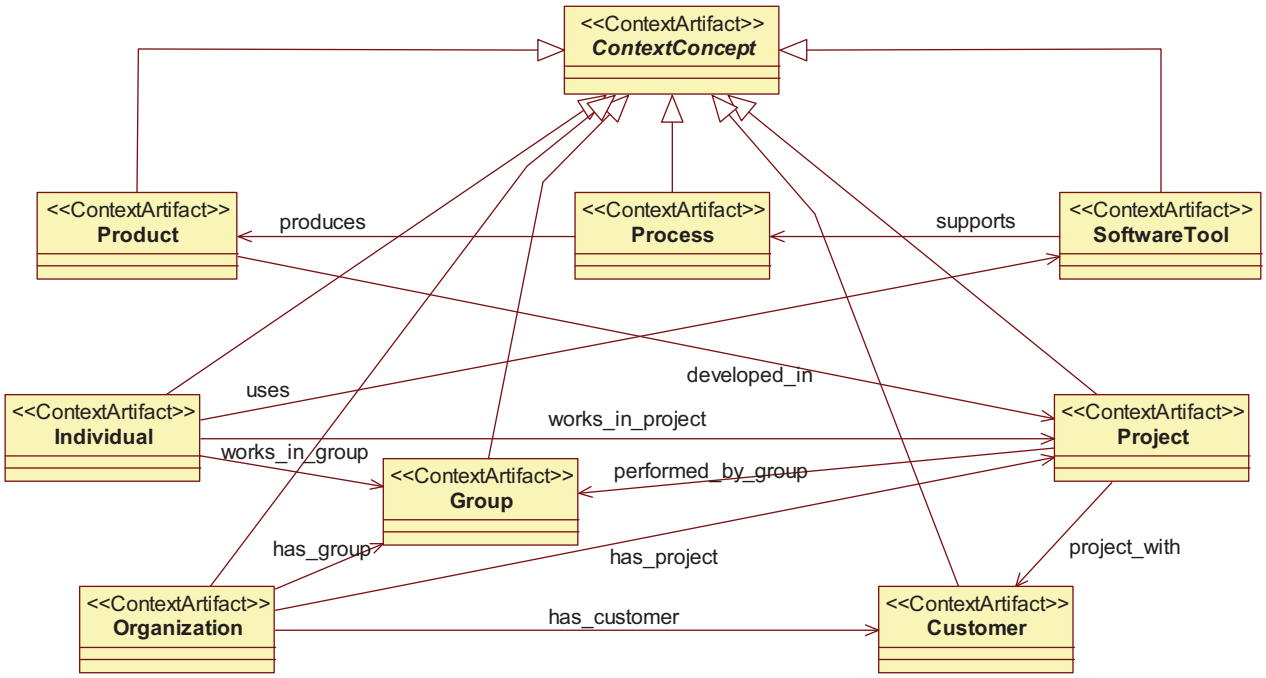


Figure 18 Context model

Context relations

Relationships occupy a leading role in the context model for the learning space approach. With the use of such relationships, a simple collection of context artifacts transforms into a structured, semantic net. The characteristics of a context or a situation can be mapped to determined roles of the related information artifacts by using such relations. In fact, a relation between two context artifacts describes a scope in which the information artifact is valid (or applicable):

- Individual *uses* SoftwareTool describes that a specific software engineer uses a specific software tool (John Doe *uses* Eclipse)
- Individual *works_in_project* Project describes that a specific software engineer is working in a specific project (e.g., John Doe *works_in_project* project_embedded)
- Individual *works_in_group* Group describes that a specific software engineer is working in a specific group (e.g., John Doe *works_in_group* department_xyz)
- Project *performed_by_group* Group describes that a specific project is done by a specific group (e.g., project_embedded *performed_by_group* department_xyz)
- Project *project_with* Customer desribes that a specific project is done for a specific customer (e.g., project_embedded *project_with* Bosch GmbH)
- Product *developed_in* Project desribes that a product is developed in a specific project (e.g., Controllersoftware_xyz *developed_in* project_embedded)
- Process *produces* Product describes that a specific process in an organization produces a specific product (e.g., process_xyz *produces* Controllersoftware_xyz)

- Software Tool *support* Process describes that a specific tool supports a specific process (e.g., Eclipse3.1.1 *supports* process_xyz)
- Organization *has_group* Group describes that a specific organization has a specific group or department (e.g., FraunhoferIESE *has_group* department_xyz)
- Organization *has_project* Project describes that a specific organization has a specific project (Fraunhofer IESE *has_project* project_embedded)
- Organization *has_customer* Customer describes that a specific organization has a specific customer (e.g., Fraunhofer IESE *has_customer* Bosch GmbH)

Definition 23 Context Model and Context Concepts

The context model has been defined in Definition 5: A *ContextModel* is used to describe a concrete software engineering situation that is determined by a finite set of context dimensions (i.e., concepts) and relations amongst these instances of these context concepts. For the learning space approach, context concepts have been selected that can be used for the adaptation of learning spaces following different learning methods; because the eight dimensions can be refined further (e.g., the individual dimension can be extended by a very detailed user model describing the competencies, preferences, learning style, etc. of a specific person), Definition 5 is now adapted to the context model in Figure 18:

$$CM \subseteq CC_{Individual} \times CC_{Group} \times CC_{Project} \times CC_{Process} \times CC_{Product} \times CC_{Organization} \times CC_{Customer} \times CC_{SoftwareTool}$$

with $CC_{Individual}$ being the set of the individual context concepts, CC_{Group} the set of the group context concepts, etc.

Definition 24 Context Concept Relations

The following relations are defined between two artifacts of two different context concepts:

$USES(i, swt) = \{swt \mid \exists i \in CC_{Individual} : i \text{ uses } swt\}$ with $swt \in CC_{SoftwareTool}$

$WORKS\_IN\_PROJECT(i, proj) = \{proj \mid \exists i \in CC_{Individual} : i \text{ works\_in\_project } proj\}$ with $proj \in CC_{Project}$

$WORKS\_IN\_GROUP(i, gr) = \{gr \mid \exists i \in CC_{Individual} : i \text{ works\_in\_group } gr\}$ with $gr \in CC_{Group}$

$PERFORMED\_BY\_GROUP(proj, gr) = \{gr \mid \exists proj \in CC_{Project}: proj \text{ performed\_by\_group } gr\}$ with $gr \in CC_{Group}$

$PROJECT\_WITH(proj, cu) = \{cu \mid \exists proj \in CC_{Project}: proj \text{ project\_with } cu\}$ with $cu \in CC_{Customer}$

$DEVELOPED\_IN(prod, proj) = \{prod \mid \exists prod \in CC_{Product}: prod \text{ developed\_in } proj\}$ with $proj \in CC_{Project}$

PRODUCES(proc, prod) = {proc | $\exists$prod$\in CC_{Product}$: proc produces prod} with proc$\in CC_{Process}$

SUPPORTS(swt, proc) = {swt | $\exists$proc$\in CC_{Process}$: swt supports proc} with proc$\in CC_{Process}$

HAS_GROUP (org, gr) = {org | $\exists$gr$\in CC_{Group}$: org has_group gr} with org $\in CC_{Organization}$

HAS_PROJECT(org, proj) = {org | $\exists$proj$\in CC_{Project}$: org has_project proj} with org $\in CC_{Organization}$

HAS_CUSTOMER (org, cu) = {org | $\exists$cu$\in CC_{Customer}$: org has_customer cu} with org $\in CC_{Organization}$

In the following, the domain model is explained, which is required for annotating learning resources (i.e., describing them with metadata) and for providing the necessary input for the adaptation process, that is, resolving the variabilities of the instructional design model and the learning resource model.

4.2.3 Domain Model

A *DomainModel* DM is used to define the so-called domain concepts of the software engineering domain and to define relations amongst instances of these sets of domain concepts (see Definition 6).

Requirements

The domain model has to fulfill the following requirements:

- Be able to formally describe common software engineering bodies of knowledge
- Be able to formally describe common process and product models and their relationships
- Be suitable for use for annotation (i.e., create metadata) of information artifacts in software engineering
- Be able to describe more complex phenomena, respectively relationships, between different types of software engineering artifacts, that match with human understanding
- Be able to support adaptation of the instructional design model and the learning resource model. Hence, serve as an input parameter for the adaptation mechanisms.
- Be as simple as possible

Underlying bodies of knowledge and reference models have been used as a basis

The context model developed in this work was derived from existing software engineering bodies of knowledge such as (*Guide to the Software Engineering Body of Knowledge, SWEBOK*, 2004) and guidelines for graduate and postgraduate education such as (Joint Task Force on Computing Curricula, 2004). In addition, reference models such as the V-Modell® XT have been studied to define the context model (Rausch et al., 2007). These extensive works cover the whole body of knowledge in software engineering, describe different ways of structuring information, and show which concept types may be used to define a body of knowledge.

Reuse model extension

The context model extends the *dependencies* characteristic of the reuse model (see also Figure 10 in Section 4.2).

Domain concepts instances

Figure 19 depicts the domain model that has been derived. It consists of five main domain concepts:

- *Individual* concept: An instance of *Individual* describes a role in software engineering (e.g., requirements engineer, quality assurer, programmer)
- *Process* concept: An instance of *Process* describes a process or activity in software engineering. This concept class also covers software engineering methods and techniques (e.g., requirements analysis, inspection, statistical testing, refactoring, etc.)
- *Product* concept: An instance of *Product* refers to an input or output created by a software engineering activity or process (e.g., non-functional requirement, test case, code, etc.). They match with the understanding of the V-Modell® XT.
- *Knowledge* concept: An instance of *Knowledge* refers to a concrete knowledge asset that is necessary to conduct a specific process (e.g., models, principles, laws, policies, etc.)
- *Software Tool* concept: An instance of *Software Tool* refers to a type of tool that supports a specific process, technique, or method (e.g., requirements analysis tool, Integrated Development Environment, etc.)

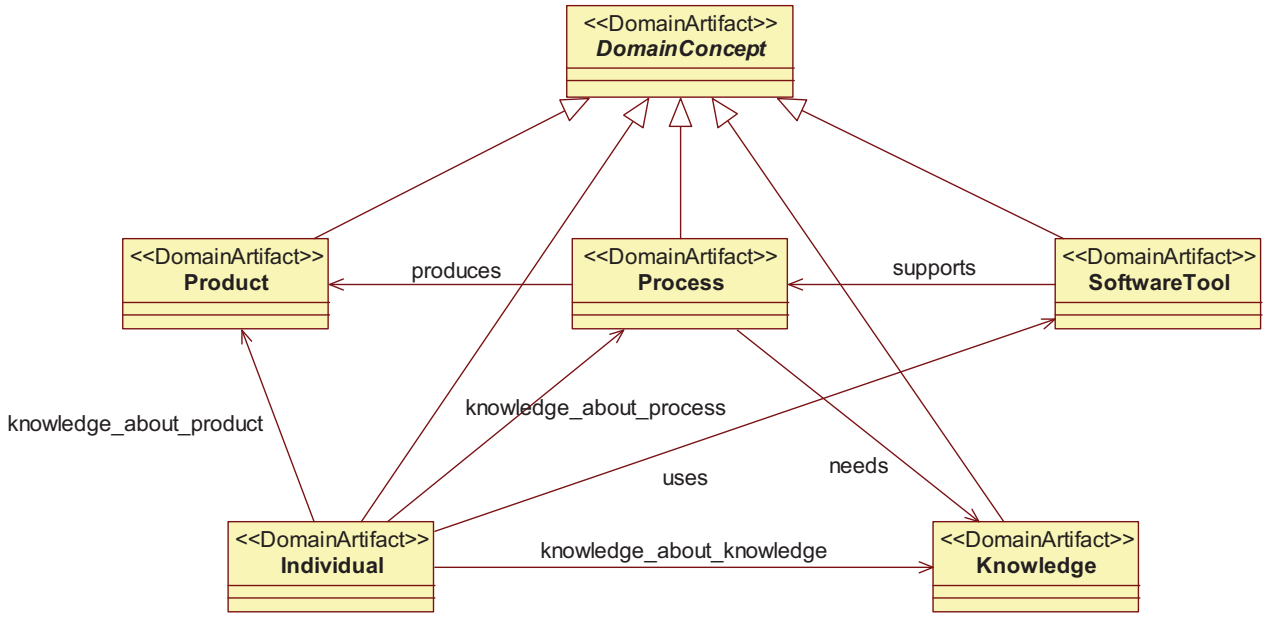


Figure 19 Domain model

Domain concepts relations

The following relations are used:

- Individual *knowledge_about_knowledge* Knowledge describes that a specific role has specific knowledge about a specific knowledge concept (e.g., project manager *knowledge_about_knowledge* COCOMO II).
- Individual *knowledge_about_process* Knowledge describes that a specific role has specific knowledge about a specific process concept (e.g., project manager *knowledge_about_process* risk management)
- Individual *knowledge_about_product* Knowledge describes that a specific role has specific knowledge about a specific product concept (e.g., tester *knowledge_about_product* statistical testing)

These three relations can be understood as part of a user model (see Section 5.4.1.1) that exactly describes the knowledge required of a specific role regarding models, principles, laws, processes, techniques, methods, products, etc.

- Process *produces* Product describes that a specific process produces a specific product (e.g., requirements elicitation *produces* non-functional requirement)
- Process *needs* Knowledge describes that a specific process needs specific knowledge to be executed (e.g., requirements elicitation *needs* quality model)
- Software Tool *support* Process describes that a specific type of tool supports a specific process or activity (e.g., issue tracker *supports* debugging)

Remark: Instances of the domain model are less specific than instances of the context model

It has to be noted that the instances of domain model concepts are not defined on the same level of abstraction as the instances of the context model: Domain model instances still refer to a class of requirements analysis processes, test tools, products, etc. Instances of the context model concepts may describe very specific versions or instantiations of a concrete product, specific organizational processes, etc. For example, in the domain model, the instance of *SoftwareTool* "Integrated Development Environment" is used instead of "Eclipse3.1.1", which can be used as an instance of the domain context concept *Software Tool*. In the following, formal definitions of the domain model concepts and their relations are given:

Definition 25 Domain Model and Domain Concepts

Domain Model – The *DomainModel* DM consists of five domain concept types:

$$DM \subseteq DC_{Individual} \times DC_{Process} \times DC_{Product} \times DC_{Knowledge} \times DC_{SoftwareTool}$$

with $DC_{Individual}$ being the set of the individual domain concept instances, $DC_{Process}$ the set of the process domain concepts instances, etc.

Definition 26 Domain Context Relations

The following relations are defined between two different domain concepts:

KNOWLEDGE_ABOUT_KNOWLEDGE(i, knw) = {knw | $\exists i \in DC_{Individual}$: i knowledge_about_knowledge knw} with $knw \in DC_{Knowledge}$

KNOWLEDGE_ABOUT_PROCESS(i, proc) = {proc | $\exists i \in DC_{Individual}$: i knowledge_about_process proc} with $proc \in DC_{Process}$

KNOWLEDGE_ABOUT_PRODUCT(i, prod) = {prod | $\exists i \in DC_{Individual}$: i knowledge_about_product prod} with $prod \in DC_{Product}$

PRODUCES(proc, prod) = { prod | $\exists proc \in DC_{Process}$: proc produces prod} with $prod \in DC_{Product}$

PROCESS_NEEDS_KNOWLEDGE(proc, knw) = { knw | $\exists proc \in DC_{Process}$: proc produces knw} with $knw \in DC_{Knowledge}$

SUPPORTS(swt, proc) = {swt | $\exists proc \in DC_{Process}$: swt supports proc} with $proc \in DC_{Process}$

The next section defines and describes a model that is essential for specifying the variabilities in the instructional design model and the learning resource model.

4.2.4 Variability Model

In order to develop context-aware learning spaces for a specific learning scenario, respectively learning method, the adaptation method can be developed by first answering the following four questions (Specht, 1998):

Question 1
What is adapted?

As defined in the previous sections, a learning space consists of learning resources and links (see Definition 16). These concepts can be adapted regarding their structure, content, and presentation. In addition, on the level of instructional design, learning goals and objectives as well as learning activities can be adapted. The following table provides a more detailed overview (the numbers in parentheses relate to examples further below):

Table 9 Adaptable concepts of the learning space approach

	Artifact	Structure	Content	Presentation
Instructional Design Model	Learning Structure Template	X (1, 2,4,5)	X	
	Learning Objective Template	X (1, 2)	X (1,2)	
	Learning Activity		X (1, 3)	
	Structure Link		X	
Learning Resource Model	Learning Page			X (6)
	Content Component	X (1, 3, 4)		X (6)
	Content Element		X (1,4)	X (6)
	Page Link		X	X (6)
	Component Link		X (5)	X (6)

Adaptation on the structure, content, and presentation levels

As can be seen from the table, concepts of the instructional design model are only adapted regarding their structure and content (e.g., the sequence (structure) of the fine-grained learning objectives in a learning structure template may change and the types of learning objectives may be adapted (content)). Links are only adapted on the content and presentation levels (e.g., the address (content) a link may refer to or its annotation (content), or its color or formatting (presentation)). On the learning resource level, the structure is only adapted for content components (e.g., the sequence of content elements can be adapted). The adaptation on the resource level focuses more on the content and presentation adaptation (e.g., the title or learning objective of a learning page may be hidden (content) or the font size (presentation) of a page, component, or a specific element may be increased).

Question 2
To what is it adapted?

What kind of information does the adaptation require? For the learning space approach, the defined context concepts sets and the domain concepts sets prescribe the possible variants of the instruction design model and the learning resource model in the scope of a specific learning scenario, respectively learning method (see Definition 7 and Definition 8).

For example, this could be a user model describing the topics learned by a learner.

Question 3
Why is it adapted?

Why is the learning space adapted to the context and domain model? The purpose depends on the aspects the learning space should improve, such as perceived usefulness of the provided learning resource, technology acceptance of the learning system, or, as in this work: understanding and application of experience packages.

Question 4
How is it adapted?

This question refers to the methods and techniques that adapt the generic artifacts to the context and domain model in the scope of a specific learning scenario. The adaptation of the learning space is done by means of a decision model. Furthermore, an overview of adaptation techniques has been given in Section 3.4.1.2.

Section 5 will answer these questions for learning spaces for experiential learning. In the following, a few examples cover the most relevant adaptations in the context of learning spaces. (These examples can be seen as possible answers to Question 2 and the main adaptations are marked in Table 9; other changes that result from the main changes are abandoned to keep the examples simple.)

Example (1) Adapting to the learning style

A learning style is used to describe individual differences in the way people learn. Kolb states in his *Learning Style Inventory* that engineers have either the style of an *Assimilator* (50%) or a *Converger* (40%) and that other styles only cover 10% (David A. Kolb, 1984). Assimilators are more interested in ideas and abstract concepts than in people and prefer logical approaches to those based on practical value. Convergers are actively experimenting and reflecting in order to build abstractions. They can solve problems and make decisions by finding solutions to questions and problems. They like more technical tasks and problems more than social or interpersonal issues. A user model of a specific software engineer describes that this person is an assimilator, which means that the structure and content of a learning space should be adapted in such a way that more abstract concepts are presented and only a few practical examples and concrete procedures for how to use the methods. Learning objectives are selected on the understand, analyze, and evaluate levels and will teach factual and conceptual knowledge during the initial stage. The different roles in software engineering can have an influence on the learning style as well.

Example (2) Adapting to the global learning goal

A learner selects a global learning goal before she accesses the learning space. The level of the global learning goal has an impact on the selection of the learning objectives. For example, if she selects the "remember" level, learning objectives of higher levels such as understand, apply, analyze, etc. should be deleted from the learning space structure template.

Example (3) Adapting to the user model

A user model may cover many characteristics of a user. It may list the known concepts of a certain domain on different levels, e.g., read, known, expert, etc. A software engineer is an expert in a refactoring technique called PreserveTempWithQuery:Process. This means, for example, that this engineer needs only a small description of this domain concept instead of a bunch of examples, exercises, and assignments.

Example (4) Adapting to time constraints

The learner selects the short variant of a learning space because he has only a short amount of time. This means that the content must be adapted. For example, the learning objectives of higher levels are deleted, since they need more time to be reached; the number of content examples is reduced, as is the amount of long descriptions elements longer than 20 lines.

Example (5) Adapting to working context

The engineer who accesses the learning space was working on a specific project and was developing a specific product that are both relevant in the scope of this learning space (e.g., because a specific technique that he wants to "understand" in the learning space was applied in that project). This context information enables the system to merge this context information with related learning activities to "remember" this situation and to connect to the concepts of the technique to be learned. In addition, component links are added (e.g., a component relates to another component that contains the project description).

Example (6) Adapting to user preferences

A learner may have set his preferences to certain types of media, such as plain text instead of formatted text, images, and non-annotated links (e.g., without tooltips). The system will transform the content into plain text, delete the images, and hide the tooltips.

In order to implement those examples, the variability model has to cope with the definition of variabilities on different levels of abstraction and their resolution during development time and run-time (see Section 3.3). It also has to cope with the usage of adaptation methods and techniques from the adaptive hypermedia domain to adapt the generic artifacts of the instructional design model and the learning resource model (see Section 3.4.1).

Requirements

The variability model has to fulfill the following requirements:

- Be able to formally describe variabilities on the instructional design model level as well as on the learning resource model level
- Be able to describe different types of variabilities (i.e., optionset, alternative, parameter, etc.) and to distinguish between "adaptive courseware generation" and "dynamic courseware generation" (Brusilovsky & Vassileva, 2003)
- Be able to formulate dependencies between variabilities in order to support traceability among different abstraction levels (i.e., deleting a type of learning activity requires deleting specific of content components)

- Be able to implement adaptation while keeping structure, content, and presentation of a learning space separated
- Separate the adaptive functionality from the content
- Be able to support adaptation during development (before the presentation of the learning space) and run-time (during the presentation of the learning space)

The variability model uses concepts from the product line approach

Variability and commonality in product line engineering are captured via *genericity*. A generic artifact is an artifact that holds all possible variants of the family, but provides some possibilities to select between them (Special Issue on Software Product Lines, 2002).

Decision model

All variabilities can be described by means of alternatives. However, as stated by Muthig, "modeling the alternatives does not define which characteristics are associated to which products, nor which dependencies and interrelationships exist among variabilities". This is captured in so-called *decision models* (Muthig, 2002). According to Muthig, a decision model structures and documents variation points, and captures relationships among variation points. Each variation point has a resolve() function that resolves the variation, i.e., chooses one to several resolutions (alternatives) from a set of choices. Figure 20 shows the variability model of the learning space approach.

Variabilities can be specified on the link and learning resource levels

Variability (see Definition 8) and commonality (see Definition 9) have been defined in Section 4.2 for both the instructional design model and the learning resource model. Hence, for the adaptation process, the variabilities of the *Link* and the *LearningResource* artifacts are relevant. This means that all artifact types of both models may have variabilities. Muthig defines generic artifacts as artifacts that contain variant elements representing a variability. Non-generic artifacts do not contain any variant elements (Muthig, 2002). Hence, generic artifacts are *Link* and *LearningResource* artifacts.

Definition 27 Generic Artifact

A generic artifact can be formally defined as a predicate:

$\forall x\ \mathrm{GENERIC}(x) \rightarrow (\exists vx_i, \exists vx_i \in VX)$ where vx_i, vx_i are two variants of x and $vx_i \neq vx_i$ and VX the set of variants of the artifact $(x \in X) \subseteq$ Link x LearningResource

Definition 28 Range

Becker uses ranges to express the scope of a variability (Becker, 2004). Four types of ranges exist: a) *BooleanRange* refers to a variability that may exist or not. It refer to a yes/no option (e.g., keep a specific learning objective or not; b) *TypedRange* refers to a variability related to a specific type that must be defined (e.g., font); c) *SelectionRange* describes a scope of a variability where more than one alternative can be chosen – multiplicity is >1 (e.g., the alternative of keeping several different types

of learning objectives may be propagated to the content resource abstraction level, hence, the alternatives are chosen based on the available content); d) *SingleSelectionRange* is a subtype of SelectionRange because only one alternative can be chosen – multiplicity is 1 (e.g., only one learning objective level can be chosen; either remember:process, understand:process, or apply:process):

$$\text{RANGE} \subseteq \text{BOOLEANRANGE} \times \text{TYPERANGE} \times \text{SELECTIONRANGE} \times \text{SINGLESELECTIONRANGE}$$

$$\forall x\ \text{BOOLEANRANGE}(x) \rightarrow x \vee \neg x$$

$\forall x$ TYPEDRANGE(x) $\rightarrow x \in [x_{rx} \ldots x_{ry}]$ where x_{rx} and x_{ry} specify the lower and upper bound of the range

$$\forall x_1, \forall x_2, \ldots, \forall x_n\ \text{SELECTIONRANGE}(x_1, x_2, \ldots, x_n) \rightarrow x_1 \vee x_2 \vee \ldots \vee x_n$$

$$\forall x_1, \forall x_2, \ldots, \forall x_n\ \text{SINGLESELECTIONRANGE}(x_1, x_2, \ldots, x_n) \rightarrow (x_1 \wedge \neg x_2 \wedge \ldots \wedge \neg x_n) \vee (\neg x_1 \wedge x_2 \wedge \ldots \wedge \neg x_n) \vee \ldots \vee (\ldots \neg x_{k-1} \wedge x_k \wedge \neg x_{k+1} \ldots) \vee \ldots \vee (\neg x_1 \wedge \neg x_2 \wedge \ldots \wedge x_n)$$

Definition 29 Variation Point

Variation points realize variabilities and are part of a generic artifact. They describe the location in a generic artifact where the adaptation will occur. The class is specialized into four subclasses of variation points that possess one of the four ranges: OptionVP, ParameterVP, OptionSetVP, and AlternativeVP.

$$\forall VP(vp) \rightarrow \exists d(d \in D)$$

where VP is the set of variation points and D is the set of decisions.

Definition 30 Decision Model

For each variation point, a decision exists. But one decision may reference more than one variation point. A variation point is only related to one generic artifact, i.e., it represents a concrete variability of a GenericArtifact in the instructional design model or the learning resource model. Decisions are variations points and also have a range. The reason for why decisions were added to the model is that decisions allow the structuring of variation points, whereas variation points represent variabilities that are resolved either during development or runtime (see Section 5.2 for the difference), but they are not related to other variation points or constrain the resolving of other variation points. They refer to one location in a generic component – this is done by means of decisions. Each decision d contains a question and related answers. After answering the questions and selecting the answers according to the defined ranges, each decision results in a resolution that resolves the variation point, e.g., chooses the alternatives.

$\exists decm\ DECM(d, res, r) \rightarrow (\forall d(d \in D) \rightarrow \exists res(res \in RES)) \wedge (\forall d(d \in D) \rightarrow \exists r(r \in RANGE))$

and $DECM \subseteq D \times RES \times RANGE$

where DECM is the set of decision models, RES the set of resolutions, and with $r \in RANGE$.

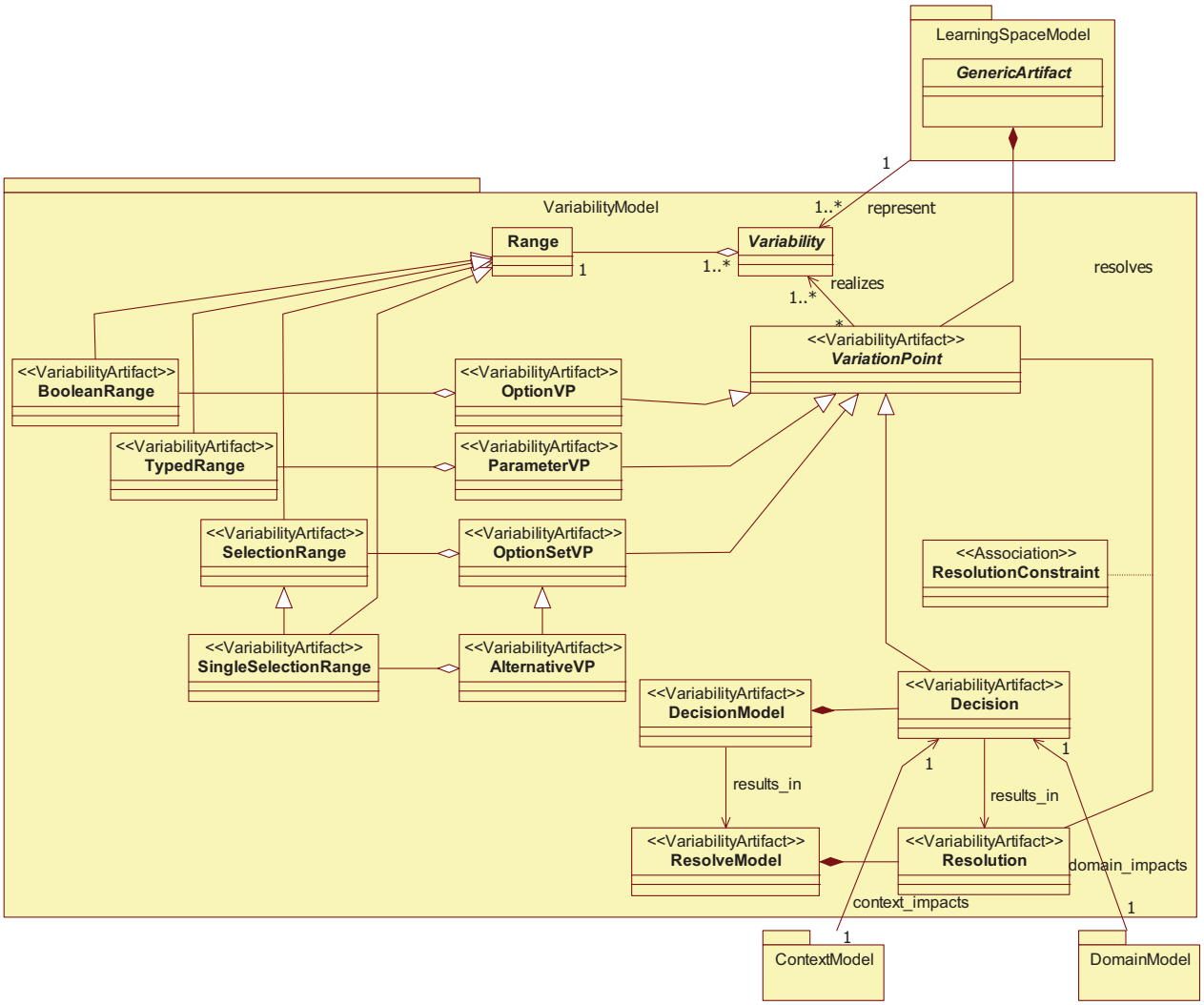


Figure 20 Variability model

The following table shows an example of a decision including the different alternatives, the generic artifact to be adapted, the type of the variation point, an informal description of its resolution, and finally the level of adaptation.

Table 10 Examples of impact indicators and their consequence for the generic artifacts

Impact Indicator (to what is it adapted?)	Value/Alternatives	Type	Generic Artifact	Type Variation Point	Possible Artifact Variants	Adap. Level
Learning activities "low level"→ preferences	• Perform TestActivity • Access "Simulation" • Perform "CollaborativeActivity"	static	Learning objective template	OptionSet	• Keep or delete TestActivity • Keep or delete Simulation • Keep or delete CollaborativeActivity	• Structure

Definition 31 esolution Model

After answering the questions and selecting the answers according to the defined ranges, each decision results in a resolution that resolves the variation point(s), e.g., chooses the alternatives.

$$\exists res\ RES(res, vp) \rightarrow \{vp \mid (\forall res(res \in RES) : res\ resolves\ vp\}$$

with RES being the set of resolutions (i.e., resolution model), and VP the set of variation points.

Definition 32 Decision

Answering a decision is done by means of the information available in a context and domain model. Hence, a concrete decision describes how specific context and domain concept instances influence a specific variability of a set of generic artifacts. The decision may have an impact on more than one variation point:

$$\forall d\ D(cm, dm, d) \rightarrow \{d \mid (\exists cm_i \in CM) \vee (\exists dm_i \in DM) : (cm_i\ impacts\ d) \vee (dm_j\ impacts\ d)\}$$

where CM is the set of context models, DM is the set of domain models, and D is the set of decisions.

Definition 33 Resolution Constraint

A so-called resolution constraint describes a relationship between a resolution of a specific decision and a "foreign" variation point (Muthig, 2002). Hence, when a resolution has a dependency on another variation point, this means that the constraint executes the resolution to this variation point even if the resolution, respectively the decision, was not directly related to that variation point:

$$\exists rescon\ RESCON(d, vp_{rel}, vp_{foreign}, res, rescon) = \{rescon \mid (\exists d \in D), (vp_{rel} \in VP), (vp_{foreign} \in VP) : RES(res, vp_{foreign})\}$$

with $vp_{rel} \neq vp_{foreign}$ and where RESCON is the set of resolution constraints, D the set of descisions, VP the set of variation point, and for RES(res, vp) refer to Definition 31.

A comprehensive example can be found in Section 5.4.3.6 and Table 25.

4.3 Comparison with other Reference Models

Brusilovsky & Vassileva (Brusilovsky & Vassileva, 2003) present three approaches of course sequencing: a course sequencing mechanism as the core of a course maintenance system for traditional, *statically sequenced courses*; *adaptive courseware generation* for generating a course suited to the needs of the students before they encounter it; and *dynamic courseware generation*, where the system observes and dynamically regenerates the course according to the student's progress.

The LS approach follows the adaptive and dynamic courseware generation approach

The learning space approach follows adaptive courseware generation because the instructional design model as well as the learning resource model are adapted to a context and domain model during development time, i.e., before the learner encounters them. However, the approach also supports dynamic courseware generation, respectively adaptation, since the system observes the context during runtime, i.e., while the learner is using it, and regenerates the learning space regarding structure, content, and presentation.

Table 11 classifies the most common reference models according to the five models of the learning space approach. The *Dexter Reference Model* for Hypertext Systems (Halasz & Schwartz, 1994) fulfills the requirement of separating structure, content, and presentation by using two interfaces: the *anchoring interface* between the storage layer and the within-component layer, and the *presentation interface* between the storage layer and the runtime layer. It can be seen from the table that the Dexter model does not explicitly model instructional design rules – they are embedded in the domain model. However, no explicit support is given for describing such rules. The reference models *Amsterdam Hypermedia Model* (AHM) (Hardman, Bulterman, & van Rossum, 1994) and *Adaptive Hypermedia Application Model* (AHAM) (Wu, Houben, & De Bra, 1998) and the *Munich Reference Model* (MRM) (Koch, 2000) use the Dexter model as the underlying model. The difference is that the AHAM model describes the pedagogical rules separately in the teaching model and that the AHM model extends the Dexter model to support multimedia content. The MRM uses the concepts of views from the *Dortmund Family of Hypermedia Models* (DFHM) (Tochtermann & Dittrich, 1996) and moves the adaptation functionality from the runtime layer to the storage layer in the so-called adaptation model. Nevertheless, the MRM is still based on the Dexter model. The MRM adds new types of links between components in the domain model in addition to the standard hypertext links: as part-of-prerequisite, variant_of, on_same_page, etc. All models but the DFHM store their content in the within-component layer.

Table 11 Comparison to other reference models in adaptive hypermedia systems

Learning Space Approach	**Dexter**	**AHAM (→ based on Dexter)**	**AHM (→ based on Dexter)**	**MRM (→ based on Dexter and storage layer of AHAM)**	**DFHM**
Instructional Design Model		(Teaching model embedded in domain model)			
Learning Resource Model	(Domain model) + Runtime Layer (instantiation of component and presentation)	(Domain model) + Runtime Layer (instantiation of component and presentation)	(Domain model) + Runtime Layer (instantiation of component and presentation for multimedia content → additional attributes such as channel and duration, etc.)	Domain Model (Hypertext with new links such as part-of-prerequisite, variant_of, on_same_page, etc.)	Links, structures, folders
Context Model	(user model)	(user model)	(user model)	(user model)	
Domain Model	Storage Layer stores information about the hypertext structure (attributes of components)	Storage Layer : domain model, user model, teaching model with pedagogical rules; the storage layer describes the actions of the adaptive engine	Storage Layer (stored as attributes of components)	Storage Layer: user model, adaptation model, domain model (concepts)	Uses hypermedia structuring concepts for organizing and categorizing hyperdocuments
Variability Model	Runtime Layer (adaptation)	Adaptive engine as part of the presentation interface	Runtime Layer (execution of adaptation)	Adaptation model by using views according to DFHM	Views, view nodes
Content Storage (Database)	Within-Component Layer	Within-Component Layer	Within-Component Layer	Within-Component Layer	

Problems of current hypermedia reference models

Hence, one problem of these reference models is that most of them do not separate instructional design structures from content/hypertext structures. The storage layer contains the domain model, which mixes up content with domain relationships and even pedagogical rules. One shortcoming of the context model is that it only contains user models, which is inadequate for adapting the hypermedia content to "situations". Variabilities of components are described by means of relationships between the variants or alternatives and not by generic artifacts. Furthermore, prerequisite links are described on the content level instead of putting them in the domain model. The adaptation is done based on rules such as construction rules, acquisition rules, and adaptation rules in MRM.

How the learning space approach addresses these problems

In order to cope with the previously stated problems, the overall conceptual model of the learning space approach separates a) instructional design structure from content, b) the description of a domain from the content, and c) the adaptive functionality from the instructional design as well as from the content. In addition, a more comprehensive context model allows adapting the learning space to more than only a user model context.

5 Learning Space Approach

"No man's knowledge here can go beyond its experience" (John Locke)

This section will instantiate the models for a real learning scenario: experiential learning in software engineering. The example is used throughout the whole chapter to explain the roles involved in the learning space approach, including their activities and the underlying techniques for adapting and generating learning spaces.

Incremental implementation of the learning space approach

The aim of the learning space methodology is not to address the whole development process first, but to focus on very specific tasks first and to extend the scope of the learning spaces later. This has the advantage that early success in important and difficult tasks, in terms of better task performance and competence development, can be used to motivate the extension and further usage of the system's scope. Especially during the requirements and programming phases, suitable tasks can be found to get the system started. The information that is necessary to choose such tasks is gathered through personal interviews with software developers and software managers, by analyzing content available in knowledge repositories, and by looking at produced artifacts such as code or software documentation.

Structure of this section

After listing the related requirements in Section 5.1, Section 5.2 presents the lifecycle of a learning space, i.e., its static adaptation, generation, presentation, and dynamic adaptation. Section 5.3 describes the experiential learning scenario that is used to explain the approach. Section 5.4 describes the different roles that are involved in creating content for learning spaces and defining the structures and variants of learning spaces, and, finally, the users who access the learning spaces. This section also covers the instantiations of the different models, i.e., the artifacts produced by the different roles. The last section explains in detail the techniques related to resolving the variation points, the static and dynamic adaptation process, as well as the techniques used for presenting the learning space.

5.1 Research Objective and Requirements

The second objective stated in Section 1 was related to the development of the techniques.

Objective 2

Objective 2: Develop techniques for the systematic, context-aware adaptation and presentation of learning spaces based on the conceptual models.

- Define a role model and describe their activities and the products it produces and consumes
- Develop a technique for the systematic and automatic resolution of static and dynamic variabilities in a learning space on the structure, content, and presentation levels
- Develop the technique in such a way that it is able to integrate resources from a technology-enhanced learning system as well as from a experience/knowledge management system

5.2 Lifecyle of a Learning Space

Four-states lifecycle model

Adaptive hypermedia systems in general generate content, present it to the user, adapt the content based on context information, and reconfigure it during interaction with the user. Jungmann and Paradies (Jungmann & Paradies, 1997) outline a four-steps lifecycle model for adaptive hypermedia systems: presentation, interaction, analysis, and synthesis. Non-adaptive systems only use the states of presentation and interaction. Furthermore, Koch extended this model, which is now adapted for the learning space (see Figure 21).

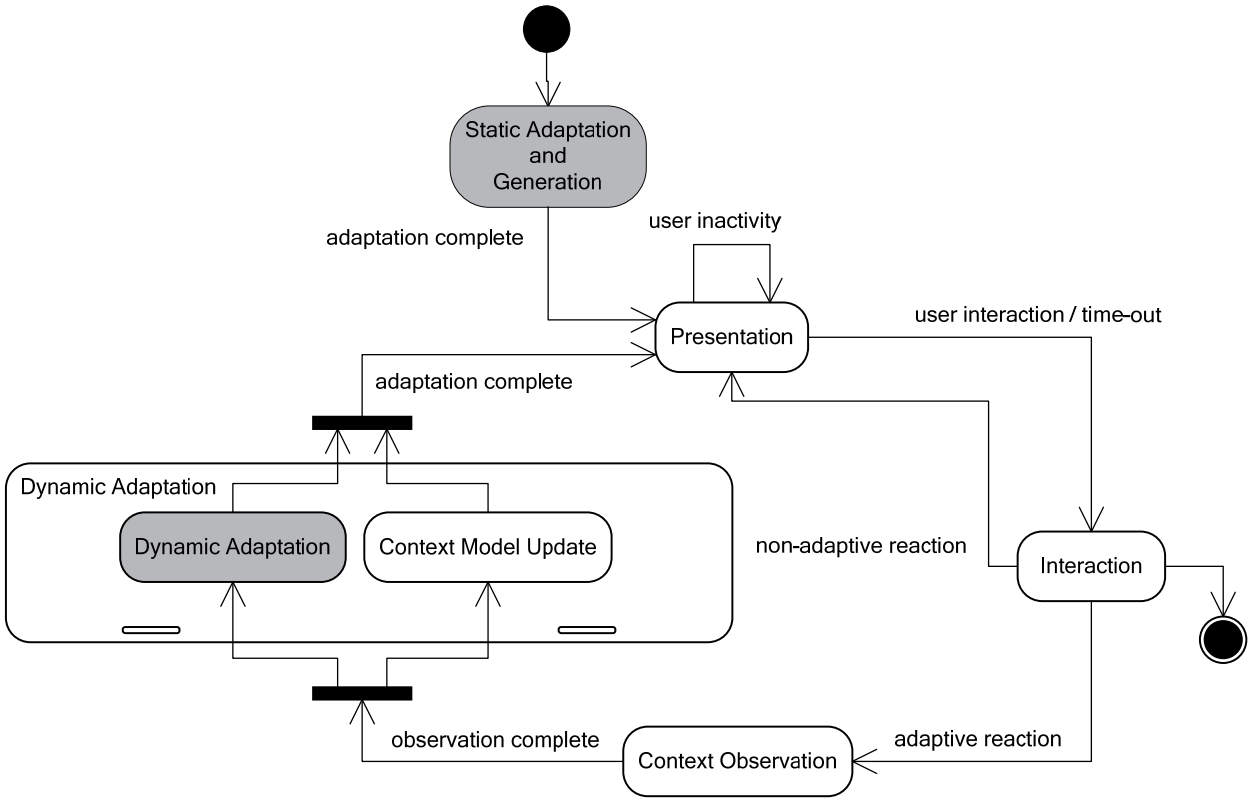


Figure 21 Lifecycle of a learning space

Static and dynamic indicators for adaptation

The model of Koch has been changed by explicitly considering adaptation during development and during runtime. The reason for this distinction is that the indicators that lead to an adaptation of the learning space can be divided into two categories (Brusilovsky, 1996): a) static or slowly changing indicators; second, b) dynamic indicators that are highly dependent on the situation and the user's behavior – this kind of indicators can alter within minutes.

Adaptation during development and runtime

An adaptive environment has to consider and manage both static and dynamic indicators. Slowly changing indicators are handled by the learning space approach before the presentation, i.e., during *development time*. In contrast, the system adapts the context model for fast changing indicators during *runtime*, i.e., during the usage of the learning space in a concrete situation. According to the definitions of adaptivity and adaptability (Oppermann, 1994), dynamic indicators are often handled by adaptive systems, i.e., adaptation is done without explicit request by the user.

Five-states lifecycle model of the learning space approach

At the beginning, the learning space is adapted based on static indicators, generated and initially presented to the user. Afterwards, the learning space runs through the states of *presentation*, *interaction*, *context observation*, and *adaptation*. The initial creation of the learning space uses parts of the context model and the domain model as static indicators to adapt the instructional design model and generate the first page of the learning space. The system remains in the presentation phase as long as the user does not interact with the learning space. User interaction may either consist of navigation activities within a learning space or changes of the user's situation (e.g., changing software engineering products such as the code in an IDE). If the interaction requires an adaptive reaction, the context is observed and an adaptation takes place.

A description of the techniques related to the different states can be found in Section 5.5.

5.3 Experiential Learning Scenario

Answer to Question 3 Why is it adapted?

This section will answer Question 3 of Section 4.2.4 "Why is the learning space adapted?" The answer to this question is related to the research question in Section 1.3: The learning space is adapted to the experiential learning scenario and the related learning method in order "to improve the *understanding* and *application* of an experience package on the one hand and *knowledge acquisition* and *perceived information quality* on the other hand".

To support the reader in understanding the roles and related activities (see Section 5.4) as well as the adaptation and generation techniques

(see Section 5.5), a scenario "Experiential Learning" is depicted in Figure 22.

Experiential learning scenario

When a software developer or a knowledge engineer documents an experience for later reuse (i.e., which is usually done by creating abstractions), he profits from being involved in the situation that leads to the experience, and from his own observation and reflection about the happening. Later, a specific situation triggers another software developer to search for experience packages (e.g., a code smell was found by a discovery tool, or a difficulty occurred during the execution of a code inspection). When experience packages can be found in the experience base, the developer evaluates the set of retrieved packages and selects one. When a software developer wants to reuse this documented experience, he or she is usually engaged in active problem-solving while reading, understanding, abstracting, or instantiating the experience package, and trying to apply the gathered knowledge to the real problem situation. To get support, the developer activates the learning space generation and selects a related global learning goal (e.g., *apply* experience package). The domain model and the context model are used to resolve the variabilities in the VariabilityModel. After the static adaptation, the learning space is generated and presented to the user. The developer accesses the information in the learning space to understand the experience package. If the interaction of the user with the learning space requires an adaptation, the VariabilityModel is again used to perform a dynamic adaptation of the learning space and probably to update the ContextModel. Afterwards, the user applies the knowledge acquired from the experience package to his situation.

The other questions of Section 4.2.4: *What is adapted? To what is it adapted? How is it adapted?* will be answered in the next sections, where the roles including their work products will be explained in the context of experiential learning. The light will be put on the shaded areas of Figure 22 because the other activities are outside the scope of this work.

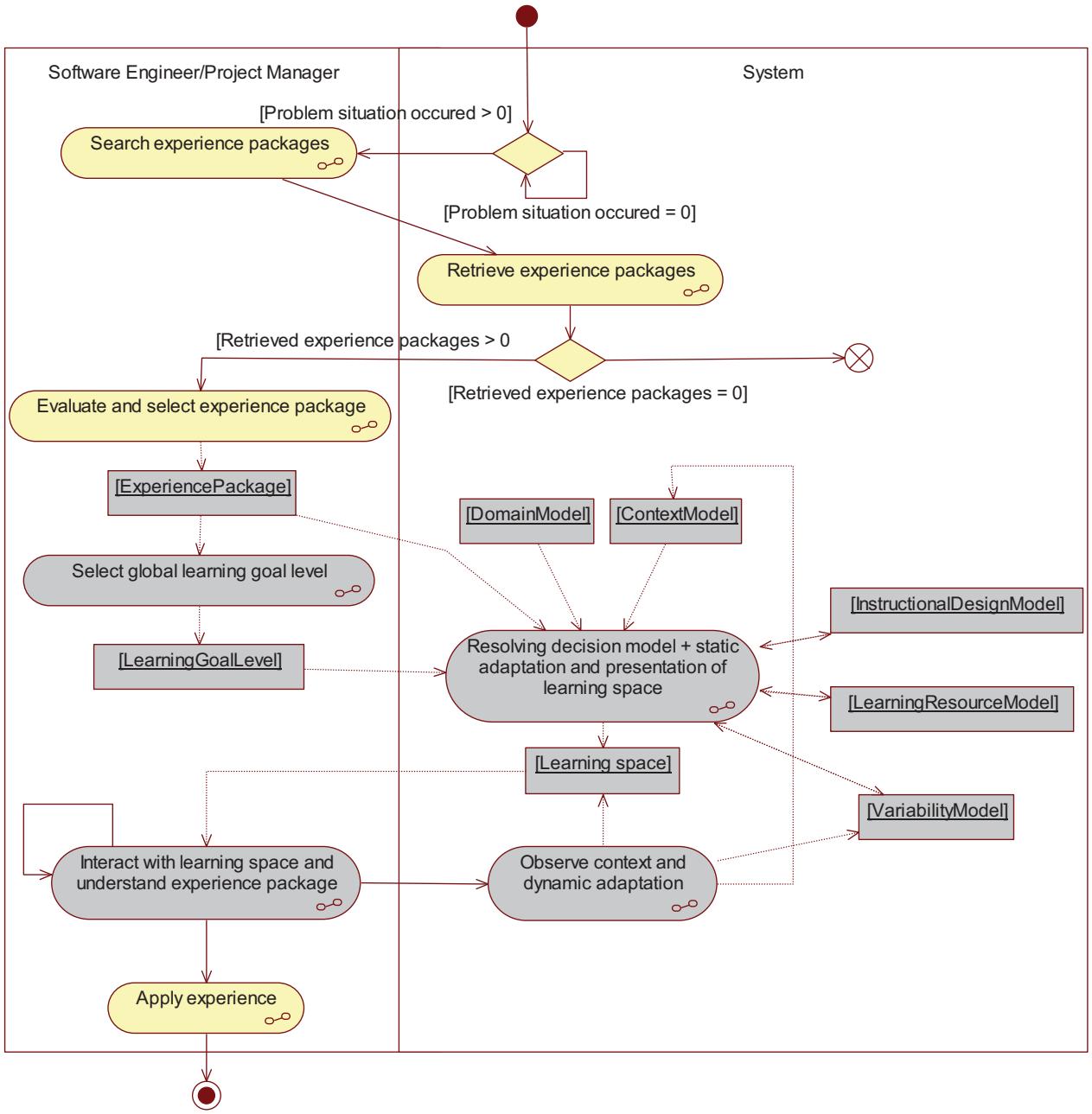


Figure 22 Activity diagram of experiential learning scenario

5.3.1 Experiential Learning Example – Experience Package

A2E structure for experience packages

Table 15 shows a shortened experience package example according to the A2E structure (Rech & Ras, 2007) as used for this work. The detailed description (D) has been left out), only the reuse model characteristics are listed.

Table 12 Experience package "Code smell comment"

	Attribute	**Value**
A. Action	Abstract	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.
	Problem	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.
	Solution	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 to improve the code: When a comment explains a block of code, you can often use the refactoring ExtractMethod to pull the block out into a separate method. The comment will often suggest a name for the new method. When a comment explains what a method does (better than the method's name), use the refactoring RenameMethod using the comment as the basis of the new name. When a comment explains preconditions, consider using the refactoring IntroduceAssertion to replace the comment with code.
B. Benefit	Effect	Improves communication. May expose duplication
C. Context	Product Context	Digital Care Giver Assistant (DGCA) 1.0
	Process Context	Agile development process
	Project Context	Open Source Practica 2007
	Individual Context	Eric Ras
	Group Context	Team of component "Interaction"
	Organization Context	Fraunhofer IESE
	Customer Context	Care center
	Software Tool Context	IDE Eclipse
D. Description	*Characteristics*	*Name:* Code Smell Comment *Function:* Remove Code Smell Comments *Use:* knowledge *Type:* qualitative experience *Granularity:* coding stage *Representation:* informal description Input/Output: Java code Dependencies: assumes person be knowledgeable in refactoring and Java programming → See Related Domain Knowledge, Related Domain Products, and Related Domain Process Application Domain: no specific *Solution Domain:* → see C. Context section *Object Quality:* see E. Evidence section
	Detailed Description	*(Left out for space reasons)*
	Related Domain Knowledge (keywords)	Code smell comment
	Related Domain Products (keywords)	Java Code
	Related Domain Process (keywords)	Extract Method, Introduce Assertion, Rename Method
	Related Domain Individual	Programmer
	Rel. Dom. Software Tool	-
E. Evidence		-

5.4 Role Model and Related Activities

What is a role in software engineering?

In the Encyclopedia Britannica, a role is introduced in sociology as describing a "comprehensive pattern of behavior that is socially recognized, providing a means of identifying and placing an individual in society" (Encyclopedia Britannica Online) Feldmann et al. describe roles as an "abstract" concept. Roles assign semantical meaning, can be hierarchically arranged and define a set of associated activities and permissions" (R. L. Feldmann, Frey, Habetz, & Mendonca, 2000). Several classifications have been defined in the past: Rombach et. al. describe roles from the perspective of software development environments in the Multi-View Process Modeling (MVP) project. They describe roles as associated tasks of one or more persons in a software project. The first type of role modifies a project's state, whereas the other roles only have observant tasks. They map persons in projects to roles and map roles to processes and activities. A role is only a "set of activities" (H. D. Rombach, Birk, Broeckers, Lott, & Verlage, 1994). Verlage defines roles as "a set of associated tasks that are assigned to one or more agents" (Verlage, 1996).

Roles for reuse

Feldman et al. distinguish two classes of roles in software development, namely, technical roles developing the software (i.e., requirements engineer, high-level design engineer, low-level design engineer, coder, verifier, system integrator, engineer, validater), and management roles for planning and managing project executions (i.e., product manager, project planner, project manager, quality assurer). They added further roles for reuse, which are also relevant for this work in the context of experience management (R. L. Feldmann, Frey, Habetz et al., 2000) (Trapp, 2002): a) the *experience manager,* who is responsible for improving and maintaining the quality of experience in the repository, b) the *experience engineer, who* is responsible for extracting experience knowledge from the projects, c) the *project supporter*, who serves as a consultant for running projects and supports project execution, d) the *librarian*, who enters data in the repository and implements the data structures, e) the *experimenter*, who designs, supervises, and draws conclusions from experiments. This classification has been detailed with a stronger support for experiments and an access control system was developed for the SFB 501 repository based on user names and the role concept (R. L. Feldmann, 1999).

The roles of the learning approach model can be mapped to the role models of Rombach et al. and Feldman et al. as shown in Table 13.

Roles for the learning space approach

The experience manager determines the structure (i.e., schema) and the content of the experience base. These activities are performed by the *competence manager* in the learning space approach. In addition to conceptually developing the context model and the domain model, he decides about the learning scenarios and opportunities for competence

development in the organization (see Section 5.4.1). The experience engineer has a lot of domain knowledge in order to package and analyze experience packages. His tasks are undertaken by the *knowledge engineer.* His main task is to extract and annotate content for learning spaces by means of the context and domain model (see Section 5.4.2). The role of the *adaptive instructional design modeler* is not covered by the other role models. His main responsibility is to develop instructional design models for certain learning scenarios and to specify variants of the learning space, i.e., to develop the variability model. Hence, the adaptive instructional design modeler must have a strong pedagogical background, knowledge about how to model variants by means of decision models, and knowledge about adaptation methods and techniques (see Section 5.4.3). The *librarian* is experienced in using data management systems and supports the competence manager and knowledge engineer in implementing the related data structures (see Section 5.4.4).

Table 13 Role model of the learning space approach

Learning Space Roles	**Technical Roles**	**Mgt. Roles**	**Reuse Roles in the Context of Experience Management**				
			experience manager	**experience engineer**	**project supporter**	**librarian**	**experimenter**
competence manager			x			x	
adaptive instructional design modeler							
knowledge engineer				x			
librarian						x	
software developer	x						
software manager		x					

The *software developer* and the *software manager* match with the technical, respectively the management, roles. They are the users of the learning space in a concrete situation. Both roles have been kept separated because they have different preferences and learning goals (e.g., a manager needs to learn in a way that he can make important decisions and a developer needs more practical knowledge that helps him not only to understand the concepts but also who him how he can apply methods and techniques to produce specific products) (see Section 5.4.5). Figure 23 shows the different roles and their activities in the context of the learning space approach by means of a use case model.

The aim of the next sections is to describe the activities of the different roles by answering guiding questions, while considering the products these activities produce and consume, which are mainly related to the instantiation of the models.

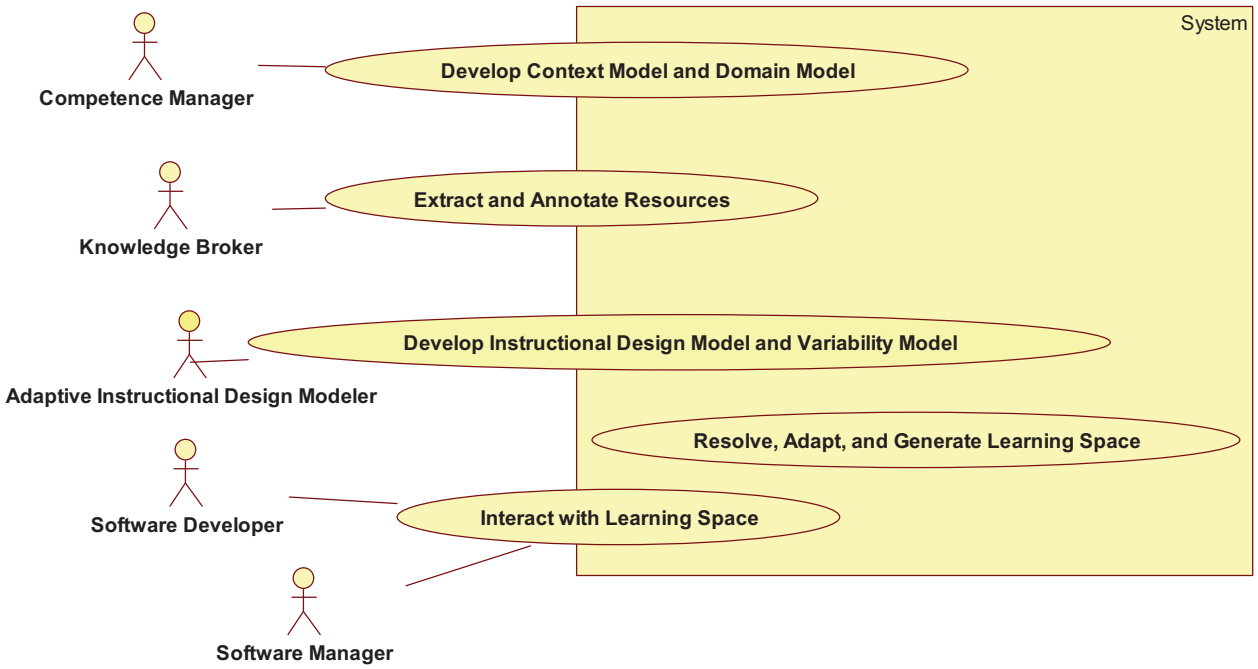


Figure 23 Roles involved in the learning space approach

5.4.1 Competence Manager

Selecting learning scenario and defining target group

The main activities of the competence manager are to decide about the learning scenarios that can be realized by the learning approach and to develop the context model (see Table 14). Together with the adaptive instructional design modeler, he selects the most promising learning scenarios and defines the target group for the learning spaces. This is done by identifying the most relevant domains for improving task performance and competence development. Selecting a domain also depends on the amount of available domain-related information in the systems. It makes no sense to choose a domain where no content is available for reuse. Learning scenarios for this approach are always close to a concrete situation during daily work, e.g., a detected problem to be solved, a small knowledge gap, etc. A concrete scenario example has been provided in Section 5.3, which describes how experience packages can be enriched by a learning space to enhance its understanding and application in the current situation. The instructional designer proposes a learning method that has to be followed by the learning spaces generated for this scenario.

Two types of triggers exist for generating learning spaces

Another important issue is to find out which situations trigger a learning need. The generation of a learning space is demand-driven, i.e., learning spaces are created based on an indicator (see Section 5.2).

Table 14 Activities of the competence manager

Question	Activity	Man./ Opt.	Product		Other Involved Roles
			Consumed	Produced	
What are the learning scenarios and opportunities for competence development in the organization?	Derive a set of learning scenarios including target group, domain, learning goal Discuss learning methods with adaptive instructional design modeler	Man.	-	List of learning scenarios, target groups, and learning methods	Adaptive instructional design modeler
What are the indicators for the need of a learning space?	Identify the situations that indicator a learning need, respectively start the generation of a learning process	Man.	List of learning scenarios and target groups	List of indicators	-
What are the available resources and relations in the repository?	Analyze the structure of the repository and derive a list of resource types and their relationships. Inform the knowledge engineer about the resource needs	Man.	Knowledge management and technology-enhanced learning system	List of resources and relationships	Knowledge engineer
What kind of resources can be used to describe situations (i.e., context)?	Identify, from the list of resources and relationships, those resources that can be used to describe situations, and develop a first ContextModel	Man.	List of resources and relationships	First draft of ContextModel	-
What context dimensions are missing and need to be added?	Extend the first draft of the ContextModel with additional context concepts and relationships. Ask support from ad. instructional design modeler for the variants of the learning spaces	Opt.	First draft of ContextModel	Initial version of ContextModel	Adaptive instructional design modeler
How can the user be stimulated to provide missing resources?	Inform the knowledge engineer about the added concepts and relationships and tell him which resources are needed to describe situations adequately	Opt.	Initial version of ContextModel	-	Knowledge engineer
How can it be implemented?	Contact the librarian to implement or extend the ContextModel	Man.	Final version of ContextModel	Implemented ContextModel	Librarian

Developing context model

The next activity of the competence manager is to look for suitable resources that may be used first to describe situations in software engineering. This is done by analyzing the concepts and relationships used in the KMS and technology-enhanced learning system (if any). He classifies these classes of resources, identifies relationships between them, and decides which resources and relationships can be used to describe a

situation. One possibility to describe this first draft of the context model is to use a class diagram as depicted in Figure 24. It can be seen that several context concepts are missing for describing a comprehensive context model as in Section 4.2.2: individual, organization, and software tool are missing. Therefore, the competence manager has to extend the model of Figure 24 with the missing concepts and relations. The final version of the context model is depicted in Figure 18.

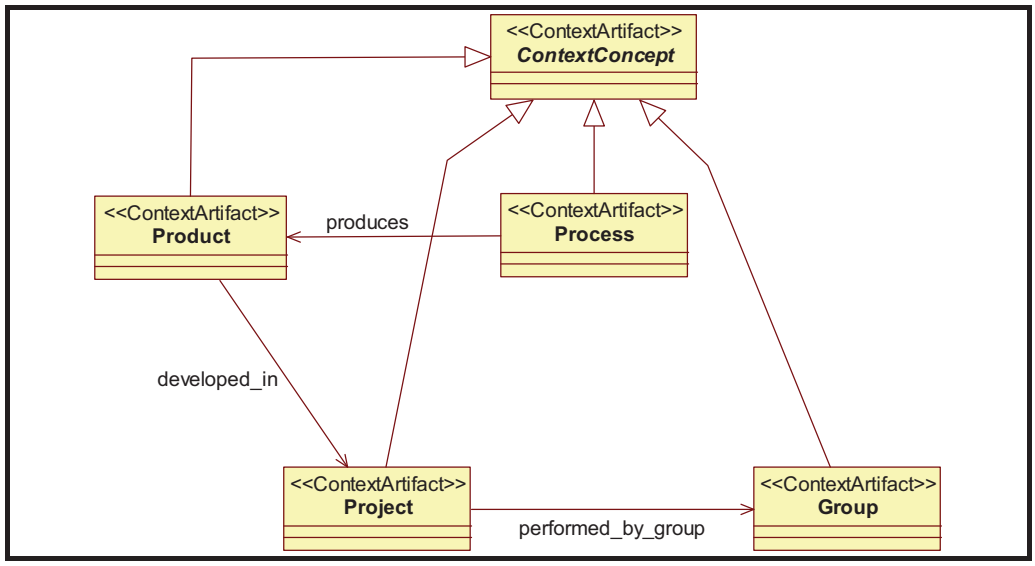


Figure 24 Example of first draft of context model

The competence manager can inform the knowledge engineer about changes to the data structure that are implemented by the librarian. This ensures that the knowledge engineer aims at providing resources to the new concepts and adding missing relationships between the resources. The learning space approach does not prescribe how a project, process, product, etc. should be formulated. The information in those descriptions will be used in a learning space as they have been documented. For example, if a learning space needs a resource that describes a specific project, then this information is reused without any changes from the KM repository. The relationships between the concepts are important because they help the users to reconstruct a situation's meaning; e.g., they may want to know with which tools and processes a product has been developed, who was the customer and who was involved in developing the product.

The user model is often missing in the context model

For the scenario of experiential learning, the adaptive instructional design modeler has to decide "what the learning space will be adapted to". Therefore, close cooperation with the competence manager is necessary, for example for defining the user model. A concept that is often missing in the first draft of the context model is the individual context concept. However, it is extremely important because it models the needs and preferences, and maybe the knowledge status of the software de-

veloper or manager who will use the learning space. Therefore, a short overview of possible dimensions is given next.

5.4.1.1 User Model

Six classes of static and dynamic indicators

There exist many different ways of defining a user model because it depends on the context it is developed for. In adaptive hypermedia, the characteristics about users can be categorized into six classes of static and dynamic indicators to adapt to (Brusilovsky, 1996). This categorization has been used as a basis for user models for adaptation in learning spaces.

Background knowledge

If a system detects that the learner does not understand a phrase or word, this fact may be the result of a lack of general knowledge or language skills (R. M. Felder & Henriques, 1995). According to the language registered in the user profile, the system could present additional information (e.g., explanation or translation). In addition, the system could provide background knowledge graded to the expertise level of the users (e.g., novice, advanced, and expert users).

Domain-specific knowledge

Domain-specific knowledge defines the knowledge about a specific domain, which is covered, for example, by a learning space. A user model could mark those concepts that have been accessed by the user. These models are sometimes called knowledge models. Domain concepts may be marked by different levels (e.g., read, known, forgotten, etc.). The way of how these levels are accessed depends on the system. Either a system chooses the level based on the time the user stays on a hypertext page, or it changes the state by analyzing the activities users perform, e.g., solving a problem or answering a question correctly.

Cognitive and affective abilities

Cognitive and affective abilities like the user's intellect, learning speed, spatial cognition, the ability to concentrate or reflect, or the motivation to learn, do have a strong impact on the learning process itself. This class of indicators may be mapped and reduced to the seven types of "Multiple Intelligence" described in (H. Gardner, 1983).

Constitutional attributes

Constitutional attributes describe physical properties of the body such as disability, age, etc., or constitutional states of the user like tiredness, concentration, etc. Age, for example, might have an impact on the selection and presentation of content (e.g., difficulty, font size).

Preferences

The learner's preferences include attributes like the preferred presentation of contents, the desired way of navigating through a learning space, the preferred learning style, or technical preferences such as bandwidth, screen resolution, etc. Preferences are partially dependent on cognitive and affective abilities as well as on constitutional attributes, which both describe what the user is able to do. Especially learning styles are often used in the preferences of a user model. Two good classifications of

learning styles, which have been used as underlying models for the learning space approach, are the *learning style inventory* of Kolb (David A. Kolb, 1984) and the *index of learning styles* of Felder and Silverman (M. Felder & Silverman, 1988).

Interests and personal targets

Interests and personal targets are important indicators for the adaptation of a learning space. These targets can be related to getting expertise for applying new methods or techniques or becoming an expert in a certain domain. These interests do not need to match with the selected learning goal of a learning space or the situation a user is currently in.

Following the insights stated in the last paragraph, the learning approach has to consider and manage static and dynamic indicators from all six classes.

5.4.1.2 Experiential Learning Example – Context Model

Advantages of describing the experience context of packages context

In this scenario, the context model has also been used to describe the context of an experience package (see Table 12). Describing the context of experiences has several advantages: context attributes help the software engineer to search for and find appropriate experiences for reuse; many of the context attributes can use a faced classification, which means that the related values are part of taxonomies (e.g., Birks faced classification for technology experience packages (A. Birk, 2001).); context categories can help the reusing person to understand, judge the applicability, and hence tailor the experience better to a new context (Rech & Ras, 2007). The following figure illustrates the instantiated context model for this example (names of relationships were left out for reasons of simplicity reasons, see Figure 18).

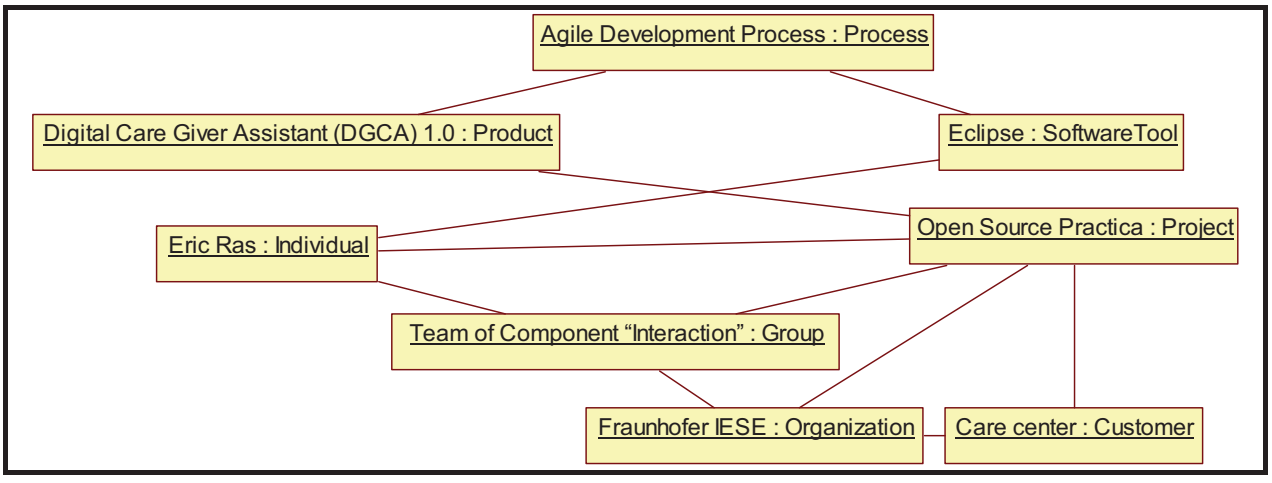


Figure 25 Example of context model for experience package

5.4.2 Knowledge Engineer

Define domain model

The two main tasks of the knowledge engineer are first, to develop the domain model with its concepts and its relationships, and second, to extract and annotate resources either from the KMS or the technology-enhanced learning system, i.e., the building blocks of a learning space. Most of the KMS and learning systems already use metadata schemas for annotating their resources. These metadata data structures need to be extended and adapted so that they can be used for generating learning spaces. This is first done by analyzing those existing metadata structures that are used to annotate resources and second, to analyze the underlying domain model (if there is one). This first draft of the domain model is then extended to develop a complete domain model for software engineering. As stated before, the knowledge engineer is very knowledgeable in the domain and should therefore be able to develop the domain model with the support of the librarian regarding technical issues. The development of the initial domain model should be discussed with the competence manager and the adaptive instructional design modeler in order to select the right domain and create relationships that are useful for adaptation.

Table 15 Activities of the knowledge engineer

Question	Activity	Man./ Opt.	Product		Other Involved Roles
			Consumed	Produced	
How are resources annotated?	Analyze the current metadata schema for annotation resources	Man.	Knowledge management and technology-enhanced learning system	Metadata schema	-
How is the domain modeled?	Analyze the data structures for annotating resources and derive a first domain model	Man.		First draft of DomainModel	-
What is the relevant domain and what domain concepts and relationships are missing?	Extend the first draft of the ContextModel with additional domain concept and relations. Ask support from the competence mgr. and the ad. instructional design modeler	Man.	First draft of DomainModel	Initial version of DomainModel	Competence manager, Adaptive instructional design modeler
How must the metadata schema be adapted?	Extend or adapt the metadata schema for resources so that the concept of the domain model can be used Ask the librarian to implement it	Man.	Metadata schema	Extended and implemented metadata schema	Librarian
How can it be implemented?	Contact the librarian to implement or extend the DomainModel	Man.	Final version of DomainModel	Implemented DomainModel	Librarian

Question	Activity	Man./ Opt.	Product		Other Involved Roles
			Consumed	Produced	
How can learning resources be extracted?	Based on the domains identified for the learning scenarios, relevant resources are identified, revised, and extracted as learning resources (i.e., learning elements)	Man.		Revised and extracted learning element	-
How can learning elements be annotated with metadata?	For each learning element extracted, provide metadata by using the domain model and store it in the database	Man.	Extracted learning element	Annotated and stored learning element	-
How can knowledge resources be annotated?	Revise existing knowledge resources in the system and add relationships based on the context model	Man.	Knowledge resource (experience packages, lessons learned, project descriptions, process descriptions, etc.)	Revised knowledge resource with additional relationships of the context model	-

Extract and annotate resources

In addition, the knowledge engineer is responsible for extracting resources from the system and classifying them either as valuable knowledge assets (e.g., lessons learned, experience packages; the role corresponds to the role in the context of experience management) or as learning elements. Both types of information need to be annotated by means of the domain model. The knowledge engineer is allowed to revise the content before storing it (e.g., shortening a description of a concrete domain concept or making a project description consistent with a project description template). It depends on the culture of the organization whether more than one knowledge engineer will extract resources. In extreme cases, for example in organizations where an open knowledge sharing culture is established, every employee may be a knowledge engineer (i.e., knowledge producer) as well as a knowledge consumer. Therefore, the idea is not to change existing policies and rules for content authoring. Learning spaces and hence the learning elements should be adapted to the existing situations, knowledge, and information within an organization.

5.4.2.1 Experiential Learning Example – Domain Model

The following figure illustrates the instantiated domain model for this example (names of relationships left out for reasons of simplicity, see Figure 19).

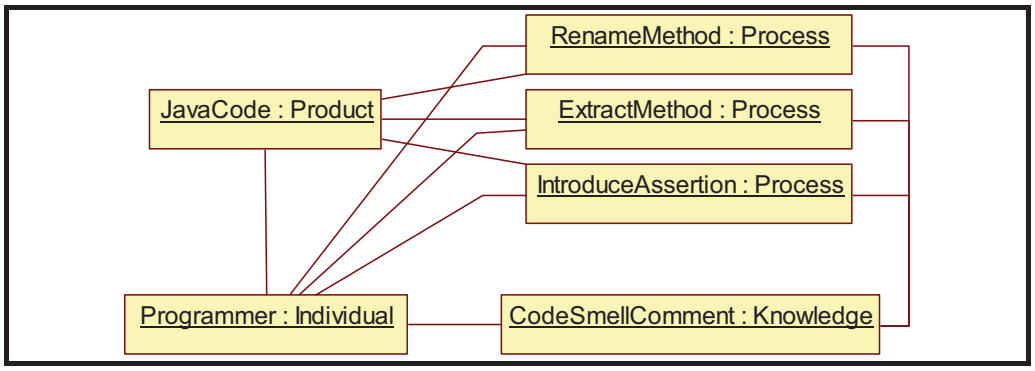


Figure 26 Example of domain model for experience package

5.4.2.2 Experiential Learning Example – Learning Elements

Common instructional design theories often speak of the following elements in the design of instruction: generalities, examples, explanations, practice items, test items, overviews, advance organizers, and analogies, among others (Yacci, 1999). For the example in this work, the learning element taxonomy of Figure 17 in Section 4.2.1.2 was used. In the following, a few examples are given including their types and the domain concept instances they are related to.

Table 16 Examples of learning elements

Domain Concept	Domain Concept Instance	Type of Learning Element	Content
Process	IntroduceAssertion	Introduction	Often sections of code work only if certain conditions are true. This may be as simple as a square root calculation working only on a positive input value. With an object, it may be assumed that at least one of a group of fields has a value in it. Such assumptions often are not stated but can only be decoded by looking through an algorithm. Sometimes the assumptions are stated with a comment. A better technique is to make the assumptions explicit by writing an assertion. An assertion is a conditional statement that is assumed to be always true. Failure of an assertion indicates a programmer error. As such, assertion failures should always result in unchecked exceptions. Assertions should never be used by other parts of the system. Indeed, assertions are usually removed for production code. It is therefore important to signal that something is an assertion. Assertions act as communication and debugging aids. In communication, they help the reader to understand the assumption the code is making. In debugging, assertions can help catch bugs closer to their origin. It has been noticed that debugging help is less important when self-testing code is written, but the value of assertions is still appreciated in communication.

Domain Concept	Domain Concept Instance	Type of Learning Element	Content
Process	RenameMethod	Description	Check to see whether the method signature is implemented by a superclass or subclass. If it is, perform these steps for each implementation. Declare a new method with the new name. Copy the old body of code over to the new name and make any alterations to fit. Compile. Change the body of the old method so that it calls the new one. If you have only a few references, you can reasonably skip this step. Compile and test. Find all references to the old method name and change them to refer to the new one. Compile and test after each change. Remove the old method. If the old method is part of the interface and you cannot remove it, leave it in place and mark it as deprecated. Compile and test.
Knowledge	CodeSmellComment	Description	Comments should be used to give overviews of code and provide additional information that is not readily available in the code itself. Comments should contain only information that is relevant to reading and understanding the program and should be added when the author realizes that something is not as clear as it could be and adds a comment. Discussion of non-trivial or non-obvious design decisions is appropriate, but avoid duplicating information that is present in (and clear from) the code. It is too easy for redundant comments to get out of date. In general, avoid any comments that are likely to get out of date as the code evolves. In addition, the frequency of comments sometimes reflects poor quality of code. When you feel compelled to add a comment, consider rewriting the code to make it clearer. Some comments are particularly helpful: - Those that tell why something is done a particular way (or why it was not) - Those that cite algorithms that are not obvious (where a simpler algorithm will not do). Other comments can be reflected just as well in the code itself! The refactorings ExtractMethod, IntroduceAssertion, RenameMethod should be used to remove this kind of code smells.

5.4.3 Adaptive Instructional Design Modeler

Serves as a consultant for many other roles

The adaptive instructional design modeler plays the most important role in the learning space approach besides the competence manager. He serves as a consultant for many other roles. Besides pedagogical knowledge about learning theories and methods, the instructional designer needs competencies in the domain of interest (but not as much as the

knowledge engineer, who can be asked). He needs competencies in technically specifying instructional designs, and must be able to reflect about all the different variants of a learning space while keeping in mind the impacting contextual indicators. One of the key activities of the adaptive instructional design modeler is to define a content elements taxonomy, and to use a domain ontology as a means for modeling the learning resource model and the related variabilities. The other activities are more related to the definition of the instructional design (see Section 5.4.3.1) and the explicit description of variabilities by means of decision models (see Section 5.4.3.5).

In general, learning spaces have two main purposes

In general, learning spaces can have two general purposes. First, a learning space can improve *short-term task performance*, i.e., by providing solutions in order to solve problems more efficiently or by offering different methods or tools that enhance a specific task. The domain (i.e., concepts/topics) under consideration is very narrow. Second, *long-term competence development* refers to learning scenarios not directly targeted at solving a problem at hand, but to learning settings where the user addresses competence gaps by using comprehensive learning spaces, which do not only cover topics related to the current working situation.

Related activities

The following table shows the activities of this role. The instructional designer assigns learning goals and learning methods to the identified learning scenarios and target group. Afterwards, each learning goal is refined into so-called learning objectives and learning activities (see examples below). Based on the context model, the instructional designer selects indicators and their impact on the learning space artifacts by listing their possible variants. The last step is related to the modeling of the decision model and related resolving operations (i.e., adaptation techniques).

Table 17 Activities of the adaptive instructional design modeler

Question	Activity	Man./ Opt.	Product		Other Involved Roles
			Consumed	Produced	
What are the learning scenarios and opportunities for competence development in the organization?	Discuss the set of learning scenarios including the target group, and the domain with the competence manager, develop learning goals, and choose learning methods	Man.	List of learning scenarios, target groups	Set of global learning goals and learning methods	Comp. manager

Question	Activity	Man./ Opt.	Product		Other Involved Roles
			Consumed	Produced	
How can the learning goal be achieved?	Break down the global learning goal by means of learning objectives	Man.	Set of global learning goals and learning methods	Set of related learning objectives for each global goal	-
How can the learning objectives be achieved?	Break down each learning objective by means of learning activities and develop a taxonomy of learning element types	Man.	Set of related learning objectives for each global goal	Set of learning activities for each learning objective (→ InstructionalDesignModel) and a learning element type taxonomy	-
What are factors that influence the learning space?	Identify the context factors that have an influence on the learning space. Discuss context model with competence manager. Identify variants of the learning space	Man.	ContextModel, InstructionalDesignModel	List of influencing indicators and variants of the learning space	Comp. manager
How can the variabilities be described?	Describe the variabilities by means of a decision model	Man.	List of influencing indicators and variants of the learning space	VariabilityModel (decision model + resolving operations)	-

5.4.3.1 Instructional Design for Learning Spaces in General

Learning has moved from a teacher-centered to a learner-centered paradigm

The philosophy of learning has started to move away from the instructional teacher-centered paradigm towards learner-centered teaching and learning practices building on socio-cognitive knowledge construction and situated learning principles. Individuals learn by developing knowledge and understanding through the forming and re-forming of concepts based on their current situation and context. "The focus of constructivism is on learners' control, with learners making decisions that match their own cognitive states and needs" (Farmer & Taylor, 2002). Today, learning is less a reaction to "being learned" more the reaction to varied requirements of learning situations and learning environments. The short innovation cycles in software engineering lead to many learning situations where new knowledge is required to solve new challenges during daily work.

Autonomic vs. directed learning

In the future, learning within an organization will balance out structured, directed learning and unstructured, autonomic learning. *Autonomic learning* consists of learning without direct teaching. Learners define their own learning goals according to given situations and select the learning steps as well as their sequence to reach the goals. Autonomic learning is more a way of explorative learning than learning based on given procedures and rules. *Directed learning* will be launched by the or-

ganization to communicate and change its strategy, culture, products, and services, which involves individuals, teams, or the entire organization. Autonomic learning originates within the organization, initiated by individuals and communities of practice.

In Section 3.4.3, it has been mentioned that a human's tasks always depend on the situation they are performed in, i.e., they are influenced by the characteristics and relationships of the *context (J. S. Brown et al., 1989)*, and that knowledge is individually constructed by following different learning paths (Jonassen, 1999).

Problem: instructional design experience is mainly shared by text

Many instructional design theories exist – even if they are difficult to apply to the learning object domain. One reason is that no explicit rules are available on how learning objects (such as learning elements in the context of this work), in general, should be selected and sequenced to make instructional sense (Knolmayer, 2003). Goodyear states that the means by which instructional design experience is shared – mainly by text – needs improvement (Goodyear, 2005). The last point makes it extremely difficult, especially for non-instructional designers, to create their learning material, since good instructional design still requires much professional experience.

5.4.3.2 Instructional Design for Experiential Learning in Particular

Reflection is crucial for experiential learning

In Section 3.1.2.3 is has been stated that experience-based learning becomes experiential learning when reflection, abstraction, and transfer of knowledge take place. Furthermore, a cornerstone of Kolb's learning cycle is the process of reflection (David A. Kolb, 1984). By reflecting about the experiences made, a learner gains new insights and competencies. A learning space must support the reflective activities *assimilation* and *accommodation,* which are interleaved (Piaget, 1976). Accommodation is a constructive reflective activity and supports the development of new concepts and schemas that arise from environmental experiences, whereas assimilation integrates new experiences into existing knowledge schemas.

Kolb's learning cycle and Merrill' first principles of instruction

The instructional design developed for the learning scenario "experiential learning" should stimulate the phases of Kolb's *Experiential Learning Circle* (i.e., making concrete experience, observation and reflection, formation of abstract concepts, and testing in new situations). It should follow Merrill's first principles of instruction (i.e., solving real-world problems, activating existing knowledge as a foundation for new knowledge, demonstration of new knowledge, application of new knowledge by the learner, integrating new knowledge into the learner's world) (Merrill, 2000).

The learning space approach should support reflection-in-action

It seems that reflection is crucial for the success of experiential learning. The value of reflection has already been proven in situated cognition theory (e.g., *Cognitive Apprenticeship* and *Anchored Instruction)*. Schön distinguishes between two types of reflection that facilitate the learning and activity of professionals: reflection-in-action and reflection-on-action (Schön, 1990, 1995). Short-term *reflection-in-action* is performed while people act and experience. The activity is reshaped while the activity is performed. *Reflection-on-action* is retrospective thinking about an experience after an activity or during an interruption. Other persons could be involved. The latter provides an understanding of practice and is a way practitioners may learn from their experience. Both reflective activities are relevant; however, reflection-in-action should be supported more by the learning space approach, since it happens directly when the experience's *element of surprise* happens (i.e., when something fails to meet our expectations) (Schön, 1990). Reflection-in-action will restructure strategies of action in software engineering and the understanding of phenomena, or change the way of framing (i.e., interpreting) problems. In addition, reflection-in-action will motivate the engineer to do on-the-spot experiments because he wants to try out and explore what he has learned immediately in the situation. Schön states that the main difference between reflection-in-action and other forms of reflection is "its immediate significance for action".

A learning space must distinguish between familiar and unfamiliar situations

One has to distinguish between familiar and unfamiliar situations in software engineering. Unfamiliar situations require a much higher amount of reflection-in-action than familiar situations because it is possible to apply rules, methods, and techniques in a routined way (i.e., executing in terms of the learning objectives of Anderson and Krathwohl (L. W. Anderson & Krathwohl, 2001). This distinction must also be made in a learning space.

Concerning the implementation of the previously stated requirements, an example will be given for the learning goal structure template and the learning activities. The knowledge types and the learning objective taxonomy, which have been described in detail in Section 3.1.2 and Section 4.2.1.1, are used for defining the artifacts of an instructional design model.

5.4.3.3 Experiential Learning Example – Learning Goal Structure Template

Underlying pedagogical models

The following decisions are made for the experiential learning scenario in Section 5.3. Here, the underlying theory is constructivism with situated cognition. The underlying pedagogical models are:

- experiential learning cycle (David A. Kolb, 1984)
- anchored instruction (Bransford, Sherwood, Hasselbring, Kinzer, & Williams, 1990)

- problem-based learning (Jonassen, 1999)
- learning-by-doing (Schank et al., 1999) as models for situated learning, and
- elaboration theory (Reigeluth, 1999).

Requirements of the learning goal structure template

All models bridge theory and practice. Experiential learning is the underlying situative pedagogical model; learning-by-doing is also a *situated* pedagogical model; anchored instruction and problem-based learning are both *cognitive* pedagogical models, and, finally, elaboration theory refers to an associative pedagogical model (see next section). The learning goal structure template should:

- Stimulate reflection-in-action
- Promote on-the-spot experimentation (i.e., during learning)
- Anchor the instruction with the situation, concrete problems, and related learning goals that are "owned" by the learner
- Represent a specific ill-defined or ill-structured problem as an example of a larger set of issues
- Support self-directed learning and self-exploration of the content
- Provide instruction that consists of experiences that facilitate knowledge construction
- Enhance cognitive flexibility as a basic principle of constructivism (i.e., by offering different perspectives on the instructional content)
- Use elaboration as a concept for instructing complex concepts, domain theories (e.g., laws, principles, etc.), and task/procedures (e.g., methods, techniques, etc.)
- Embed the concepts into real-world stories in order to enhance problem building (i.e., construction) and problem solving

Classification of the instructional strategy

According to the comparison framework for instructional strategies by Reigeluth and Moore (Reigeluth & Moore, 1999), this strategy for supporting experiential learning can be classified as:

Table 18 Example of learning goals and related learning objectives

Framework	Learning space for experiential learning
Type of Learning	Focus on *understanding relationships* and applying *generic skills*
Control of Learning	Focus on *student-centered* control
Focus of Learning	Focus on learning *domain-specific topics* and *problem-oriented* learning
Grouping of Learning	Focus more on *individual* and less on *group* learning
Interaction of Learning	Focus on *student<-> tool, student<-> information,* and *student<-> environment*
Support for Learning	Focus on *cognitive support*

Based on the learning scenario (see Section 5.3), the adaptive instructional design modeler defines three learning goals and related learning objectives (see Section 4.2.1.1 for their formal definitions).

Table 19 Example of learning goals and related learning objectives

Learning Goal (according to (L. W. Anderson & Krathwohl, 2001))	**Learning Objectives**
Remember experience package	• Remember the domain knowledge concept(s) from this experience package • Remember the domain product concept(s) from this experience package • Remember the domain process concept(s) from this experience package • Remember the domain individual concept(s) from this experience package • Remember the domain software tool concept(s) from this experience package • Remember the situation describing the context from this experience package • Summative Self-Assessment: Remember experience package
Understand experience package	• Understand the domain knowledge concept(s) this experience package • Understand the domain product concept(s) from this experience package • Understand the domain process concept(s) to from this experience package • Understand the domain individual concept(s) from this experience package • Understand the domain software tool concept(s) from this experience package • Understand the situation describing the context of this experience package • Summative Self-Assessment: Understand experience package
Apply experience package	• Understand the domain knowledge concept(s) from this experience package • Understand the domain product concept(s) from this experience package • Understand the domain individual concept(s) from this experience package • Apply the domain software tool concept(s) from this experience package • Apply the domain process concept(s) from this experience package • Understand the situation describing the context of this experience package • Summative Self-Assessment: Apply experience package

As can be seen from the previous table, the adaptive instructional design modeler decides to have a sequential structure for the learning objectives, each referring to a certain domain concept (or several concepts) that is related to the experience package. The overall learning goal is assessed by a summative test activity for each of the learning objectives. The next section shows how each learning objective can be refined by learning activities.

5.4.3.4 Experiential Learning Example – Learning Objective Templates

The following decisions are made for the experiential learning scenario in Section 5.3. The adaptive instructional design has to decide which content element types are available in the repositories, respectively which types of content may be extracted from the KMS by the knowledge engineer. The following table shows the selected instructional and situational content elements of this learning scenario:

Table 20 Examples of instructional and situational content elements

Instructional Content Elements	**Description**
Learning objective	States the selected overall learning objective
Definition	Provides a definition of a domain concept instance
Description	Provides a more detailed description of a domain concept instance
Example	Provides an example of a domain concept instance
Experience	Provides the detailed description of an experience package
Scenario	Explains a domain concept instance by means of a practical scenario
Exercise	Shows an exercise that can be solved by the learner
Simulation	Illustrates a domain concept instance by a simulation
Collaborative activity	Provides a contact to a knowledgeable colleague
Domain Relation	Shows a relationship between two domain concepts instances
Integrated Practice Activity	Performs a practice activity in one's own working environment
Situational Content Elements	**Description**
Product	Provides a description of the product of the relevant situation
Process	Provides a description of the process of the relevant situation
Project	Provides a description of the project of the relevant situation
Individual	Provides a description of the individual of the relevant situation
Group	Provides a description of the group of the relevant situation
Organization	Provides a description of the organization of the relevant situation
Customer	Provides a description of the customer of the relevant situation
Software Tool	Provides a description of the software tool of the relevant situation
Context Relation	Shows a relationship between two context concept instances

Based on these content element types, learning activities can be specified for each learning objective listed before. Each learning activity is described by a tupel (activity, contentElementType). The sequences of the learning activities support the previously stated requirements for the learning goal structure template. Some of the requirements are addressed on the next lower abstraction level (Section 5.4.3.5).

Table 21 Examples of learning activities for the learning goal "remember"

Learning Goal	**Learning Activities**
Remember the domain knowledge concept(s)	• Read concept *description* • Read concept *example*
Remember the domain product concept(s)	• Read concept *description* • Read concept *example*
Remember the domain process concept(s)	• Read concept *description*
Remember the domain individual concept(s)	• Read concept *description*

Learning Goal	Learning Activities
Remember the domain software tool concept(s)	• Read concept *description*
Remember the situation	• Read *experience* (i.e., read the complete experience package) • Read *product* • Read *process* • Read *individual* • Read *group* • Read *organization* • Read *project* • Read *customer* • Read *softwareTool*
Summative Self-Assessment: Remember experience package	• Read learningObjective • Read *domainRelation*: "Knowledge x is needed for using process y for product z" • Read *domainRelation*: "Individual x has knowledge about product y" • Read *domainRelation*: "SoftwareTool x support process y"

Table 22 Examples of learning activities for the learning goal "understand"

Learning Goal	Learning Activities
Understand the domain knowledge concept(s)	• Read concept *definition* • Read concept *description* • Read concept *example* • Read *domainRelation*: "Knowledge x is needed for using process y for product z"
Understand the domain product concept(s)	• Read concept *definition* • Read concept *description* • Read concept *example* • Read *domainRelation*: "Process x produces product y"
Understand the domain process concept(s)	• Read concept *definition* • Read concept *description* • Read *domainRelation*: "Process x is supported by softwareTool y"
Understand the domain individual concept(s)	• Read concept *description* • Read *domainRelation*: "Individual x has knowledge about process process y" • Read *domainRelation*: "Individual x has knowledge about product product y" • Read *domainRelation*: "Individual x has knowledge about software tool softwareTool y"
Understand the domain software tool concept(s)	• Read concept *description*
Understand the situation	• Read *experience* (i.e., read the complete experience package) • Read *product* • Read *process* • Read *individual* • Read *group* • Read *organization* • Read *project* • Read *customer* • Read *softwareTool* • Read *contextRelation*: "Process x produces product y" • Read *contextRelation*: "Product x developedin project y"

Learning Goal	Learning Activities
	• Read *contextRelation*: "Individual x works in project project y" • Read *contextRelation*: "Individual x works in group group y" • Read *contextRelation*: "Organization x has project project y"
Summative Self-Assessment: Understand experience package	• Read *learningObjective* • Perform *testActivity* "What were the most important concepts in the experience package" → Recall • Perform *testActivity* "Illustrate how the experience package concepts are related" → Exemplify • Perform *testActivity* "Explain the cause and effect when this experience is applied to your situation" • Perform *testActivity* "Summarize in your own words what the experience package is about" • Perform *testActivity* "Compare your situation with the situation of the experience package" • Perform *testActivity* "Compare your problem with the problem stated in the experience package"

Table 23 Examples of learning activities for the learning goal "apply"

Learning Goal	Learning Activities
Understand the domain knowledge concept(s)	• Read concept *definition* • Read concept *description* • Read concept *example* • Read concept *counterExample* • Read *domainRelation*: "Knowledge x is needed for using process y for product z"
Understand the domain product concept(s)	• Read concept *definition* • Read concept *description* • Read concept *example* • Read concept *counterExample* • Read *domainRelation*: "Process x produces product y"
Understand the domain individual concept(s)	• Read concept *description* • Read *domainRelation*: "Individual x has knowledge about process process y" • Read *domainRelation*: "Individual x has knowledge about product product y" • Read *domainRelation*: "Individual x has knowledge about software tool softwareTool y"
Apply the domain software tool concept(s)	• Read concept *description* • Perform integratedPracticeActivity "Use the tool"
Understand the situation	• Read *experience* (i.e., read the complete experience package, especially the probably ill-defined problem) • Read *product* • Read *process* • Read *individual* • Read *group* • Read *organization* • Read *project* • Read *customer* • Read *softwareTool* • Read *contextRelation*: "Process x produces product y" • Read *contextRelation*: "Product x developed in project y" • Read *contextRelation*: "Individual x works in project project y" • Read *contextRelation*: "Individual x works in group group y" • Read *contextRelation*: "Organization x has project project y"

Learning Goal	Learning Activities
	• Organizing "Structure your situation. What is the problem?"
Apply the domain process concept(s)	• Read concept *definition* • Read concept *description* • Read concept *example* • Read *domainRelation*: "Process x is supported by softwareTool y" • Read concept *scenario* • Read concept *observation* • Perform *collaborativeActivity* (e.g., contact knowledgeable colleague) • Access concept *simulation* • Perform concept *exercise*
Summative Self-Assessment: Apply experience package	• Read *learningObjective* • Perform *testActivity* "What were the most important concepts in the experience package" → Recall • Perform *testActivity* "Illustrate how the experience package concepts are related" → Exemplify • Perform *testActivity* "Explain the cause and effect when this experience is applied to your situation" • Perform *testActivity* "Summarize in your own words what the experience package is about" • Perform *testActivity* "Compare your situation with the situation of the experience package" • Perform *testActivity* "Compare your problem with the problem stated in the experience package" • Perform *integratedPracticeActivity* "Use the process in your situation" • Perform *testActivity* "Compare your solution with the solution stated in the experience package" • Perform *testActivity* "Conclude your solution and write down an observation"

The following section shows how these learning activities are realized by learning pages, which consist of content components and content elements, by first providing a list of underlying theories.

5.4.3.5 Experiential Learning Example – Content Components and Content Elements

Elaboration theory is used on the structure and content levels

First, the elaboration theory (Reigeluth, 1999) was used as the underlying theory for structuring learning spaces on the component and element level because it belongs to the learner-centered instructional design strategies and helps to select and sequence content in a way that it optimizes the attainment of the learning objectives. Reigeluth distinguishes between conceptual, theoretical, and simplifying conditions elaboration sequences. Hence, the sequence of content elements may depend a) on relationships between the different topics (i.e., the relationships between the domain concepts), b) on the topics themselves (i.e., each domain concept is taught one by one until the required depth of understanding is reached), or c) on spirals (i.e., the learner masters a topic gradually in several passes; first the basics of each topics and then

the details). Interrelationships between topics may be learned more easily by spiral sequencing, hence topical and spiral sequencing should be seen as a continuum and not as two separate sequencing techniques.

- *Conceptual elaboration* follows the fact that people store concepts under a broader, more inclusive concept in their cognitive structure: present the easiest, inclusive, most familiar organizing concepts that an engineer has not yet learned first, and then proceed to narrower, less inclusive concepts. Conceptual elaboration can be realized by topical or spiral sequencing.

  → Conceptual elaboration is used for the learning space product, individual, and software tool domain concept instances
- *Theoretical elaboration* is suitable for learning spaces that focus on a set of interrelated principles, which are usually elaborations of each other. These principles are interrelated by causal relationships among the changes of specific concepts and exist in a broader and narrower sense (i.e., separation of concerns in general, or separation of concerns on the design level). Unlike concepts, the broader principles are easier to learn than the narrower ones. Ausubel states that a principle is stored under a broader, more inclusive one in the cognitive structures. Again, this elaboration technique starts with the principles on the highest level and progresses to the narrower principles or provides more details about a principle (e.g., What else happens? When does this cause have this effect? Why and which way do things change? How much do they change?). Theoretical elaboration can be realized by topical or spiral sequencing and move from the simple to the complex.

  → Theoretical elaboration is used for learning space knowledge domain concept instances
- The *simplifying conditions method* first teaches the easiest and simplest version of a task or procedure and then progressively more complex versions of it by making the learner aware of the difference between the different versions, respectively complexity levels, of the task. The steps should be presented in order of their performance. This is different from the hierarchical sequencing approach (i.e., subskills are taught first, and the main skills related to a task/procedure are taught at the end). The advantage is that the learner gets the whole picture of the task from the beginning by offering complete real-world realizations.

  → Simplifying conditions method is used for learning space *process* domain concept instances

Mayer's approach is used on the presentation level

Second, the engineer should be involved in three cognitive processes (Mayer, 1999): attending the relevant information (i.e., selecting), mentally organizing the information into a coherent mental representation (i.e., organizing), and integrating the information with existing knowledge (i.e., integrating). Mayer's approach is used on the presentation level of the content:

- *Selecting* information can be supported by using highlighting (e.g., headings, italics, bold face, arrows, icons, underlining, margin notes, repetition, white space, and captions), instructional objectives, and summaries.
- *Organization* of information can be supported by using outlines (e.g., comparison/contrasts structure, classification structure, enumeration, generalization and cause-effect structures), signaling headings, pointer words, structured illustrations, and coherent text structures.
- *Integration* of information is fostered by using advanced organizers, captioned multiframe illustrations, narrated animations, worked-out examples, and elaborative questions (Mayer, 1999).

Gardner's Approach for Understanding is used the on structure and content levels

Third, Gardner presents in his *Multiple Approach for Understanding* the concepts for selecting significant topics, using so-called entry points, and gives analogies and examples (H. E. Gardner, 1999). Relevant entry points that could engage the learner in the topics for this work are:

- telling stories,
- using quantitative patterns and statistics,
- active engagement through hands-on activities, and
- social interaction by collaborative arrangement and providing group settings.

Telling analogies places a learner in the center of a disciplinary topic, stimulates his interests, and ensures cognitive commitment for further exploration.

Because of the limited amount of space, only a few examples for learning activities → learning components/element transformations are given.

Example: Learning goal

Figure 27 shows the example that has already been used for other explanations in earlier chapters. This example shows the learning structure template and the other learning space artifacts for the overall learning goal remember code smell comment:experience package with seven learning objectives as listed in Table 21.

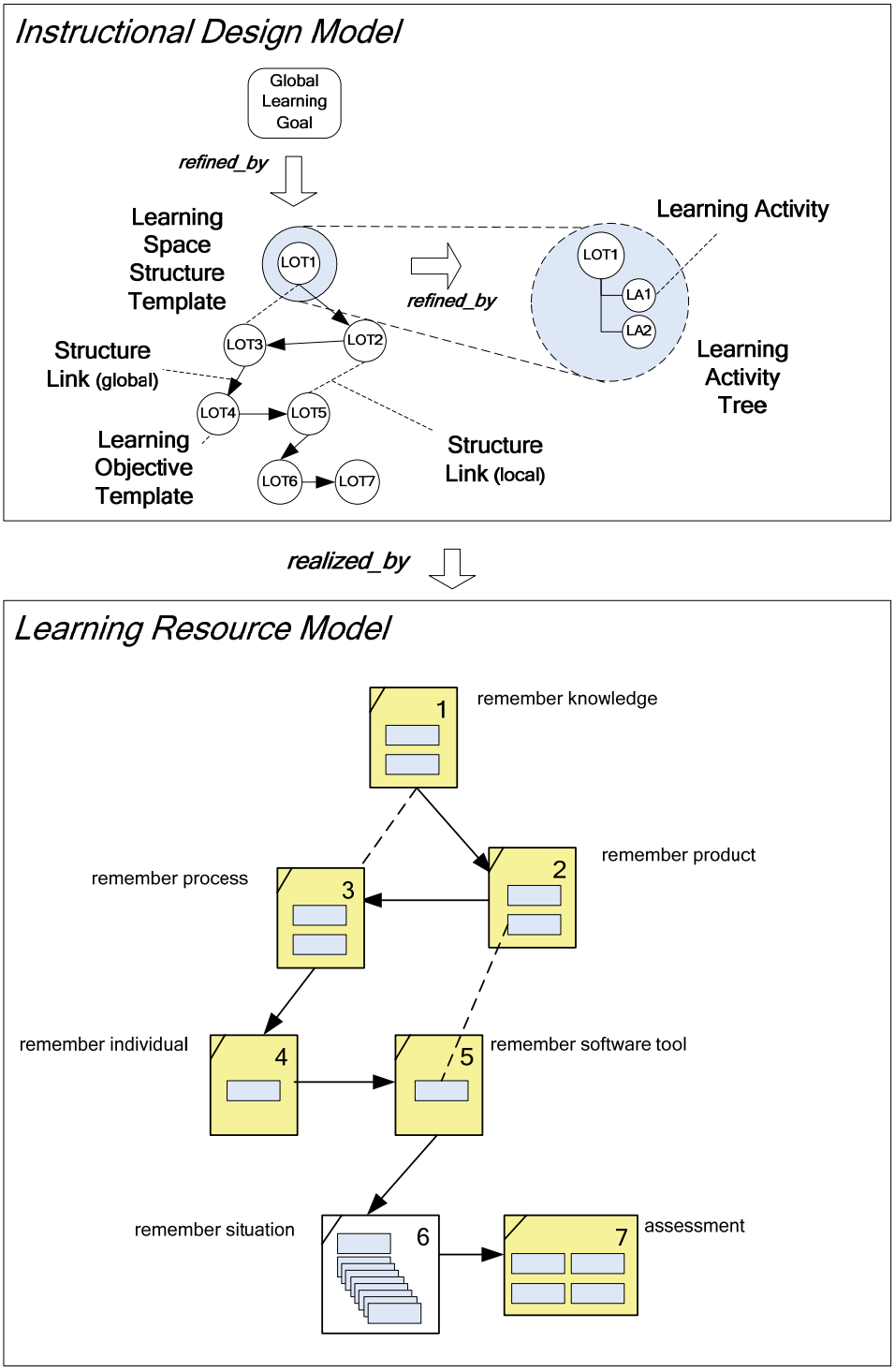


Figure 27 Examples of learning pages and content components of the types *description* and *example*

Example: Learning objective template

The following example shows the realization of LOT3 *remember the domain process concepts,* i.e., domain concepts that are related to the experience package. In this case, the experience is related to the process concepts rename method, extract method, and introduce assertion. In the example, the two related learning activities are simply realized by one component by following the elaboration theory: first, an overview of all processes is given by means of three learning elements of the type description. Afterwards, examples of each process are given.

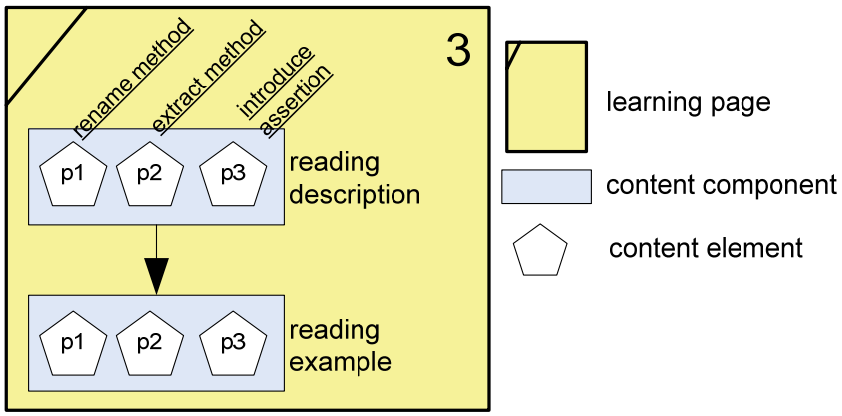


Figure 28 Examples of content component and related content elements

Implementations (i.e., presentations) of the learning components and elements can be found in Section 6.

5.4.3.6 Experiential Learning Example – Variability Model

The first activity of the adaptive instructional design modeler is to identify together with the competence manager a set of indicators that impact the variability model. Indicators are directly related to the context model. Table 24 shows an example of such an indicator list developed by the instructional designer. Each indicator is described by means of its location in the context model (see Section 5.4.1.2), possible values, its type (i.e., static/dynamic; see Section 5.2), the generic artifact(s) it refers to, the type of variation point (see Section 4.2.4), possible adaptations to the generic artifact(s), and, finally, the level of adaptation (i.e., structure, content, presentation; see Section 4.2.4).

Table 24 Examples of impact indicators and their consequence on the generic artifacts

ID	Impact Indicator (to what is it adapted?)	Value	Ind. type	Generic Artifact	Type variation point	Possible Artifact Variants	Adaptation Level
D1	Learning style → context_preferences	• Assimilator • Converger	static	Learning objective template	Alternative	• Provide abstract instructional element types (definition, description), delete CollaborativeActivity, integratedPracticeActivity, simulation • Provide more concrete types (example, exercise, simulation) and abstract concepts to the end	• Structure
D2	Learning activities "top level"→ context_preferences	• Read "learning objective" • Read "learning content" • Read "situational content"	static	Learning objective template	OptionSet	• Keep or delete learning objective • Keep or delete learning content • Keep or delete situational content	• Structure
D3	Learning activities "low level"→ context_preferences	• Perform TestActivity • Access "Simulation" • Perform "CollaborativeActivity"	static	Learning objective template	OptionSet	• Keep or delete TestActivity • Keep or delete Simulation • Keep or delete CollaborativeActivity	• Structure

ID	Impact Indicator (to what is it adapted?)	Value	Ind. type	Generic Artifact	Type variation point	Possible Artifact Variants	Adaptation Level
D4	Time → context_preferences	• t < 5min • 5min ≤ t < 15min • 15min ≤ t	static	learning goal structure template, learning objective template, content component	Alternative	• No assessment LOT, delete definition in general; only one example per concept, first concrete then more abstract, omit CE longer than 20 lines • No individual LA, max 3 examples per CC • No action; keep all content	• Structure • Content
D5	Knowledge model → domain specific knowledge	• Domain concept instance not accessed • Domain concept instance accessed	static / dyn.	Content Element	Alternative	• Don't mark CE • Mark CE as "read"	• Presentation
D6	Context experience package	• Learner is author • Learner worked in same project • Learner worked in same group	static	Learning goal structure template	OptionSet	• Only show remember level • Remember first situation and the content • Remember first situation and the content; ask colleague	• Structure • Structure • Content
D7	Global learning goal	• GLG=remember/underst/apply	static	Learning goal structure template	Alternative	• Select LGST "remember" • Select LGST "understand" • Select LGST "apply"	• Structure
D8	Font size → context_preferences	• 4 < f_size <32	static /dyn.	Learning page, content component, content element	Parameter	• Change font size	• Presentation
D9	Headings→ context_preferences	• heading=yes/no		Learning page, content component, content element	Alternative	• Use headings • don't use headings	• Presentation

ID	Impact Indicator (to what is it adapted?)	Value	Ind. type	Generic Artifact	Type variation point	Possible Artifact Variants	Adaptation Level
D10	Guidance → context_preferences	• Local guidance=yes/no •	static	Structure link/page link component link	Alternative	• Use local guidance • Don't use local guidance	• Structure
D11	Guidance → context_preferences	• Global guidance	static	Content component	Alternative	• Use overview CC • Don't use overview CC	• Content
D12	Resolution → context_preferences	• Different resolutions	static	Learning page, content component	Parameter	• Max 3 CE per learning page	• Presentation
D13	Language → Background	• Different languages	static	Learning page, content component, content element	Alternative	• Present content according to selected language	• Presentation
D14	Mediatype → context_preferences	• Select preferred mediatypes	static	Content element	OptionSet	• Present only content element of the selected type(s)	• Content
D15	Domain knowledge, individual, product, processes, software tool → Personal interests	• Preferred domain concept instances (i.e., topics) where the learner wants to acquire competencies independent of the selected learning space	static	Content component	Parameter	• Present additional content components related to the topics of interest	• Content

Decisions and their resolutions

As explained in Section 4.2.4 and Figure 20, decisions, which are variation points, result in a concrete resolution when they are resolved. The resolution of a concrete decision may have an impact on other variation points. The following table describes a few variabilities of Table 24 by means of the decisions, the related queries to resolve them, and the constraints.

Table 25 Example of a decision model and related constraints

ID	Query of Decision	Type of Variation Point	Choices (query results)	Constraint	Adaptation Operation
D2	select (top_level_types) from context_preferences	OptionSet	1. Learning_objective 2. Instructional_content 3. Situational_content	2(No): exclude 3	1(yes):Present_LA(learning_objective) else exclude 2(yes): Present_LA (instructional_content) else exclude 3(yes): Present_LA (situational_content) else exclude
D3	select (low_level_types) from context_preferences	OptionSet	1. TestActivity 2. Simulation 3. CollaborativeActivity	-	1(yes):present_CC(TestActivity) else exclude 2(yes): present_CC(Simulation) else exclude 3(yes): present_CC(CollaborativeActivity) else exclude
D4	select(time) from context_preferences	Alternative	1. t < 5min 2. 5min ≤ t < 15min 3. 15min ≤ t	1(yes): part_resolve D3.1(no)	1(yes):exclude_CC(definition) present(max=1, CC_example) sequence(illustration, fundamental) present(length=20, CE_all) 2(yes):exclude_DC(individual) present(max=1, CC_example) 3(yes): nop

Example: Contraints

Decision D2 queries the preferred content types on the top level, i.e, on the learning objective level. Since the type of the variation point is an *OptionSet*, more than one choice can be selected by the learner. Each choice (i.e., answer to the decision) results in an adaptation operation. For example, when the learner select "yes" for "learning objective", then the learning object will be presented. When the learner selects "no" for instructional content, this resolution has a constraint that excludes decision D3. Another type of constraint only impacts part of a variation point, i.e., a choice. For example when the learner prefers to get short learning spaces with an expected learning time less than five minutes, this requires that "test activities" are excluded from a learning space and that choice 1 of decision D3 is resolved with "no" in the resolution model.

Section 5.5 will describe the generation process of a learning space in detail and will elaborate the resolve and adaptation operations.

5.4.4 Librarian

The activities of the librarian have a more technical nature, because the librarian supports the other roles by implementing their models, taxonomies, metadata vocabularies, data structures in repositories, etc. He helps the other roles to use specific tools for developing their products.

Table 26 Activities of the librarian

Question	Activity	Man./ Opt.	Product		Other Involved Roles
			Consumed	Produced	
How can a domain model be implemented?	Implement the context model	Man.	ContextModel	Implemented ContextModel	Competence manager
How can a context model be implemented?	Implement the domain model	Man.	DomainModel	Implemented DomainModel	Knowledge engineer
How can a metadata vocabulary for knowledge resources be implemented?	Implement the metadata vocabulary	Man.	MetadataVocabulary	MetadataVocabulary	Knowledge engineer
How can resources be stored?	Develop data structures for storing resources	Man.		Data structures	Knowledge engineer

5.4.5 Software Developer and Software Manager

Software developer and software manager have different learning needs

The software developer and the software manager are the two main types of people who will access and use a learning space in a working situation. Their working activities are different – the activities related to the learning space approach are not different. Both roles may document software engineering artifacts such as products, processes, customers, projects, etc. The knowledge engineer can help the software developer and the software manager to annotate the artifacts or to use relationships of the context model. The other activity is related to access to and interaction with the learning space. Especially the interaction with the learning space (e.g., accessing the learning pages, reading learning elements, following links, solving assignments, etc.) and interactions within working situations are important, since they may ask for dynamic adaptations of the learning space or the user model (see Section 5.2).

Table 27 Activities of the software developer and software manager

Question	Activity	Man./ Opt.	Product		Other Involved Roles
			Consumed	Produced	
How can knowledge be documented?	Document knowledge by using the concepts of the context model	Opt..	-	Knowledge resource, context model	Knowledge engineer
How can knowledge resources be annotated?	Add relationships based on the context model to the knowledge resource	Opt.	Knowledge resource	Knowledge resource with additional relationships of context model	-
How can I use a learning space?	Access a learning space and interact with it in a specific situation by following the instructions	Opt.	Learning space	Interaction activities	-

5.5 Learning Space Generation Techniques

Main steps for generating a learning space

After presenting the different roles as well as the related activities and the products they consume and produce, this section concentrates on the description of techniques for:

1. *resolving a decision model* – the resolution technique uses the models developed by the different roles in the previous sections to resolve the decisions of the decision model. The technique produces a so-called resolve model, which consists of operations for adaptation (1 in Figure 29).

2. *statically adapting and presenting the learning space* – the static adaptation technique (i.e., adaptation during development time) executes the operations of the resolve model and adapts the learning space on the structure, content, and presentation levels. Then, the instructional design model artifacts are instantiated by searching for appropriate content (2 in Figure 29).

3. *observing context and dynamically adapting the learning space* – the dynamical adaptation technique (i.e., adaptation during run-time) adapts the learning space based on interactions between the user and the learning space or based on other contextual observations (3 in Figure 29).

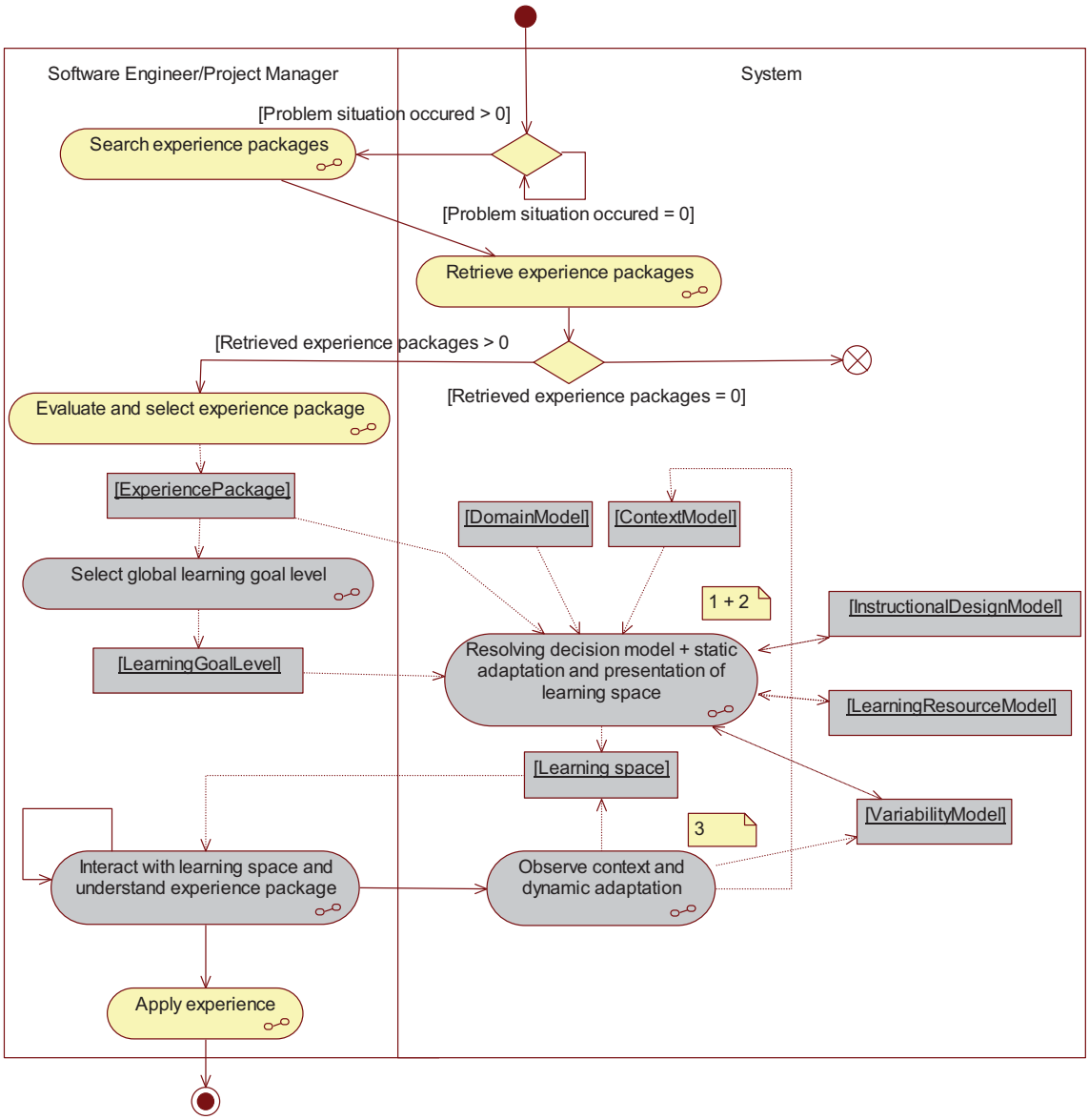


Figure 29 Relevant techniques of the learning space approach

These techniques will be elaborated in the next sections by using the following template, which is also used to specify important subfunctions:

Table 28 Template for describing a technique or function

Name	Name of the technique or function
Short description	A short description of the technique or function
Initiating Event	Description of the initiating events that trigger the start of the process
Stopping Event	Description of the stopping events that trigger the end of the process.
Preconditions	Conditions that need to be met before the process can be started.
Postconditions	Conditions that need to be met before the process has stopped.
Input Data	Documents, data, or other input needed during the execution of the process.
Output Data	Documents, data, or other output created during the execution of the process.

Pseudo code In addition to these tables, pseudo-code is used to describe some of the algorithms, since pseudo code is an environment-independent description of the key principles of an algorithm (Dalbey, 2003). Details about data initialization procedures and data structure conversions (e.g., creating a DOM model from the DTD schemas or database tables from the OWL ontology) can be found in (Ilin, 2008).

5.5.1 Resolution, Static Adaptation, and Presentation Technique

The purpose of this technique is to resolve the variation points in the variability model. This is done by performing several queries on the context and domain models. The resulting resolve model contains all the resolutions of the decisions and hence, adapts the generic artifacts by calling their `adapt()` functions. Finally, the present function generates the content artifacts of the learning space. Figure 30 illustrates the high-level steps of the resolution and static adaptation techniques.

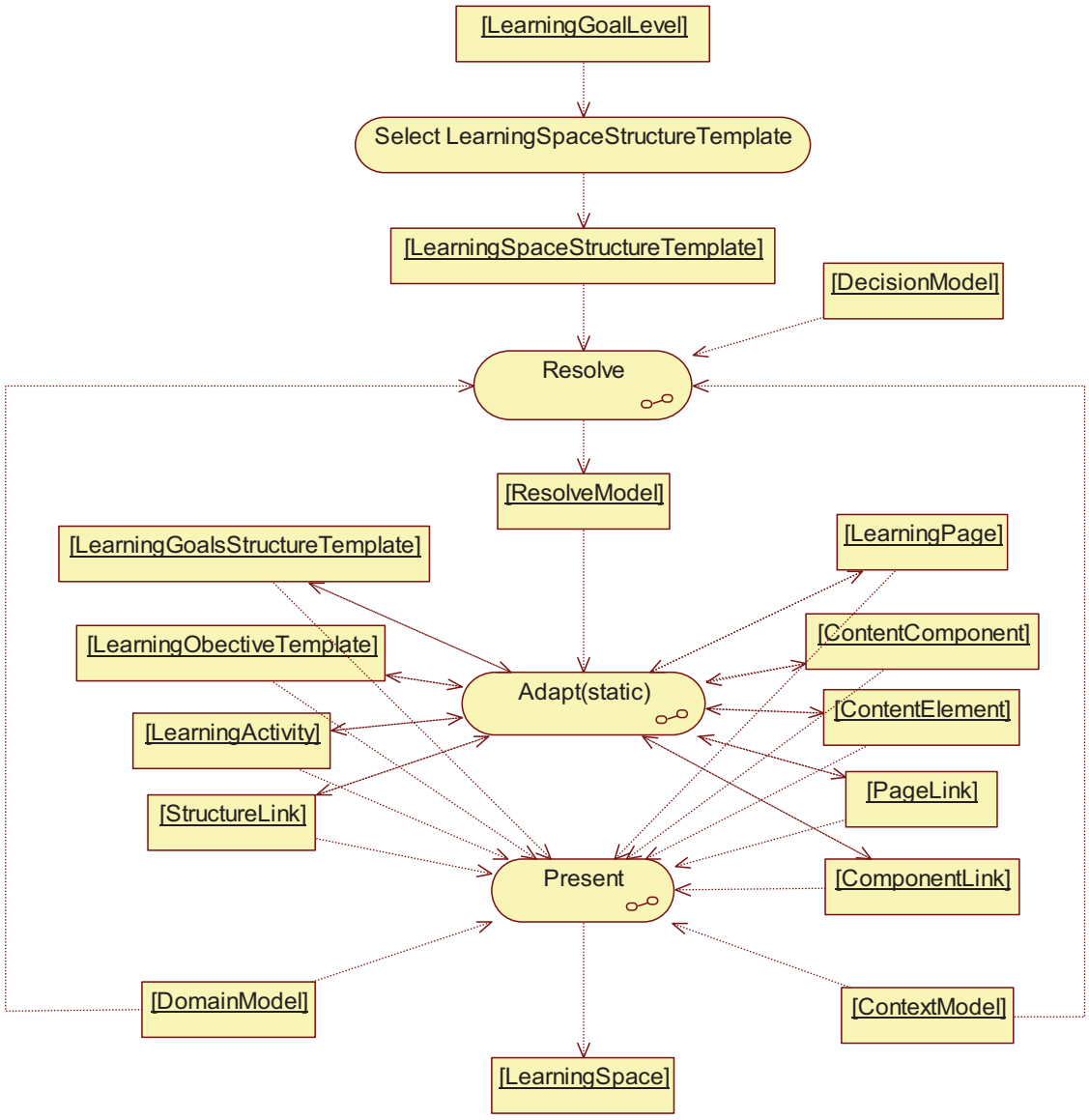


Figure 30 Relevant techniques of static adaptation and presentation

Resolving

After selecting a learning goal level (which is done by the software developer/manager in the experiential learning scenario), a `LearningSpaceStructureTemplate` is retrieved by the system. This template has a dependency on a concrete `DecisionModel`, which is used for resolving the variation point (i.e., decision).

Table 29 Specification of resolve() of VariationPoint

Name	resolve()
Short description	The resolve function performs several queries on the context and domain models to resolve the static variation points. While doing this, resolution constraints are considered.
Initiating Event	Learning space generation activated by user or system
Stopping Event	All variation points (i.e., decisions in the decision model) are resolved
Preconditions	LearningGoalLevel selected
Postconditions	No decisions open
Input Data	LearningSpaceStructureTemplate, DecisionModel
Output Data	ResolveModel

The next table explains the principle algorithm of `resolve()`.

Table 30 Pseudo-code of function resolve()

resolve()

```
resolve(){
   FOR each decision in decisionModel where type=static DO
      result = performquery(decision.query)
      IF decision TYPEEQUAL paramVP DO
         intialize(decision, result)
         ELSE deletechoice(decision, result)
 update(resolveModel)
      ENDIF
      IF resolution has resolution contraint DO
         resolve (variationPoint) //recursion
      ENDIF
   ENDFOR
   store(resolveModel)
}
```

Adaptation

The resolution is done by means of the query result, which determines which choices of a decision should be deleted or initiated in the case of a variation point of the type ParamVP. The `ResolveModel` is then used to statically adapt the generic learning space artifacts. This is done by calling the `adapt()` function of the generic artifacts.

Table 31 Specification of adapt() of GenericArtifact

Name	adapt()
Short description	The adapt function uses the resolve model adapt to the generic artifacts
Initiating Event	Resolve() terminated
Stopping Event	All generic artifacts adapted to the context
Preconditions	ResolveModel available where all decisions have been resolved
Postconditions	All adaptations successful and all variabilities resolved
Input Data	LearningSpaceStructureTemplate, LearningObjectiveTemplate, LearningActivity, StructureLink, LearningPage, ContentComponent, ContentElement, PageLink, ComponentLink
Output Data	Adapted generic artifacts

The next table explains the principle algorithm of `adapt()`.The adaptation depends on the selected adaptation technique chosen by the adaptive instructional design modeler. For the experiential learning scenario, adaptation techniques according to the adaptation level in Figure 29 were chosen. Describing the detailed algorithms for adapting the generic artifacts would be outside the scope of this work.

Table 32 Pseudo-code of function adapt()

adapt()

```
adapt(){
   FOR each resolution in resolutionModel DO
         staticadapt(resolution.genericartifact, resolu-
           tion.adapttech)
   ENDFOR
}
```

Presentation

Finally, the present function of each learning space artifact of the first learning objective is called to retrieve suitable content elements (i.e., either instructional or situational) and to construct the first learning page by means of content components, content elements, page links, and component links. The subsequent learning pages are created dynamically during runtime when the learner accesses them by using a link (either a page link or a component link).

Table 33 Specification of present() of LearningResource and Link

Name	present()
Short description	The present function instantiates instructional design artifacts that have been requested by the learner and retrieves situational as well as learning content that should be presented elements from the database
Initiating Event	Adapt() terminated
Stopping Event	none
Preconditions	All adaptations successful and all variabilities resolved
Postconditions	All artifacts of the learning page presented successfully
Input Data	LearningSpaceStructureTemplate, LearningObjectiveTemplate, LearningActivity, StructureLink, LearningPage, ContentComponent, ContentElement, PageLink, ComponentLink, DomainModel, ContextModel
Output Data	Presented LearningPage consisting of ContentComponent(s), ComponentElements, PageLink(s), and ComponentLink(s)

The selection of content is done based on information from the domain and context models. This depends on the learning scenario to be implemented. Therefore, the following descriptions are kept generic and do not depend on a specific learning scenario. For example, for the experiential learning scenario, the input for the queries comes from the context description of the selected experience package. However, other learning scenarios may provide this input from the working environment (e.g., an IDE).

The next table explains the principle algorithm of `present()`.

Table 34 Pseudo-code of function present() of LearningResource and Link

present()

```
present(){
   next=1
   FOR next LearningObjectiveTemplate DO
        FOR each LearningActivity DO
          dom_result = retrieve domain input for query
          //e.g., from experience package
          con_result = retrieve context input for query
          //e.g., from experience package
          keyworddom = query domainontology()
          keywordcon = query contextontology()
          initiate related contentComponent (keyworddom, key-
          wordcon)
          FOR each contentComponent DO
            retrieve contentElement(keyworddom, keywordcon)
            present contentElement()
          ENDFOR
          IF ANY
            present componentLink()
          ENDIF
          Present pageLink() // link to other learning pages
          respectively learning objectives
          WAITUNTIL interaction from user //the next page is
          presented on demand
          INCREMENT next
        ENDFOR
   ENDFOR
}
```

5.5.2 Context Observation and Dynamic Adaptation Technique

The previous section has described the static adaptation of a learning space. A variation point has a type, either static or dynamic. As presented in the lifecycle model in Figure 21, dynamic variabilities are resolved during runtime, i.e., while the learner accesses the learning space and works in his environment. This adaptation is triggered by an interaction of the learner by means of a so-called adaptation event by the system. Such an adaptation event leads to an observation (e.g., a query to the knowledge model of the user model or a system-based diagnosis of code smells). The result of such an observation is a dataset that describes relevant characteristics of a situation. These characteristics determine the adaptation of the learning space. Dynamic adaptation is done on the learning resource artifacts and not on the instructional design artifacts.

Figure 31 illustrates the high-level steps of the context observation and dynamic adaptation technique.

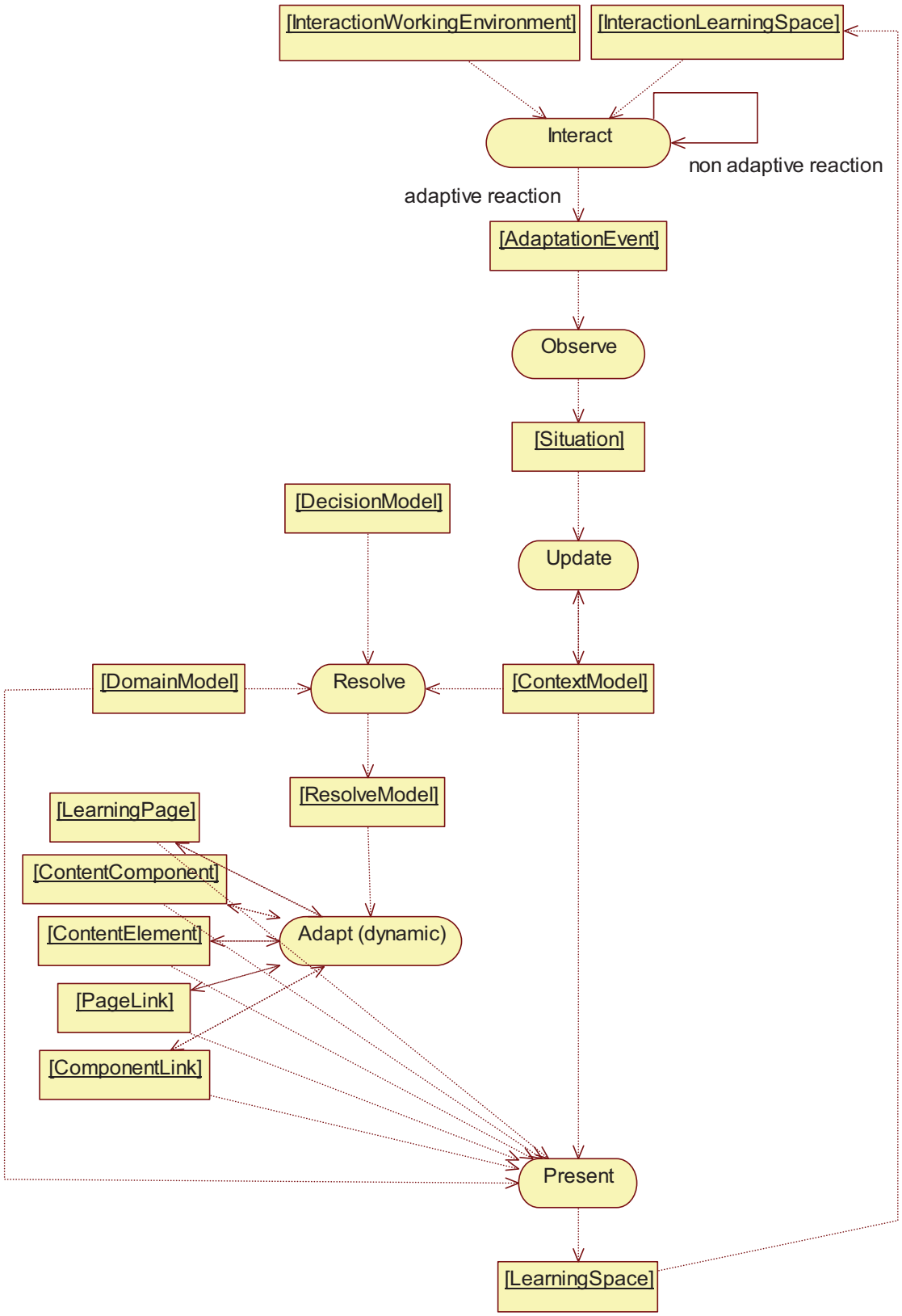


Figure 31 Relevant techniques of context observation and dynamic adaptation

Interaction After accessing the first page of the learning space, the learner interacts

with the learning space and his working environment. Each interaction is checked by the interaction function as to whether the system should react in a non-adaptive manner or whether it should adapt the learning space.

Table 35 Specification of interact()

Name	interact()
Short description	The interact function decides whether an interaction requires an adaptation or not
Initiating Event	Learning space accessed by the user
Stopping Event	None, continuous
Preconditions	User interacts with learning space of working environment
Postconditions	Decision made about adaptation or not
Input Data	InteractionLearningSpace, InteractionWorkingEnvironment
Output Data	Null or AdaptationEvent

The next table explains the principle algorithm of `interact()`.

Table 36 Pseudo-code of function interact()

interact()

```
interact(){
   access eventBase
   FOR each interaction DO
      IF interaction is adaptive event DO
         generate adaptiveEvent
      ELSE NOP
   ENDFOR
}
```

Observation

The function `interaction()` accesses an `eventBase` that contains all possible monitored events in the learning space of the working environment. When an `interaction` is marked as adaptive event in `eventBase`, then an `adaptiveEvent` artifact is created. Such an artifact specifies what needs to be observed during the next step. The following steps describe the `observe()` function.

Table 37 Specification of observe()

Name	observe()
Short description	The observe function observes the current situation and creates a situation object that specifies the characteristics of a situation
Initiating Event	Initiate function returns an adaptiveEvent
Stopping Event	None
Preconditions	AdaptiveEvent available
Postconditions	Situation object created
Input Data	AdaptiveEvent
Output Data	Situation

The next table explains the principle algorithm of `observe()`.

Table 38 Pseudo-code of function observe()

observe()

observe(){
FOR each context attribute of adaptationEvent DO
situation.attribute = retrieveSituationattri-
bute(adaptationEvent.attribute) // retrieve current
context
ENDFOR}

Update

For each `adaptationEvent` that specifies the context characteristics to be retrieved, queries are performed to retrieve the current context. For example, a learner reads a specific content element, which is related to a specific domain concept instance "refactoring". An adaptiveEvent specifies that the time for reading this learning page must be accessed. This value is used to resolve a dynamic variability. For example, when the time for reading this page has exceeded five minutes, then this domain instance is marked as "read" in the user's knowledge model of the context model by using the function `update()`.

Table 39 Specification of update()

Name	update()
Short description	The update function updates the context model before the learning resource artifacts are adapted
Initiating Event	Observe function and Situation object
Stopping Event	None
Preconditions	Situation object available
Postconditions	contextModel updated
Input Data	Situation, contextModel
Output Data	contextModel

The next table explains the principle algorithm of `update()`.

Table 40 Pseudo-code of function update()

update()

update(){
FOR each attribute of situation DO
change contextModel(situation.attribute)
ENDFOR
store(contextModel)
}

The other steps of the context observation and the dynamic adaptation technique are the same as for static adaptation, except that variation points with the type *dynamic* are resolved and that only artifacts of the learning resource model are adapted during runtime. Therefore, these steps are not described again.

6 Learning Space Tools

> "*Wiki:* The simplest online database that could possibly work" *(Leuf & Cunningham, 2001)*

Web 2.0 concepts (e.g., collaboration, sharing), features (e.g., tagging, folksonomies), and tools (e.g., Wikis, blogs) support quick and easy sharing of knowledge as well as creation of learning content in a software organization (S. Weber et al., 2008). Web 2.0 refers to a class of Web-based applications that harness collective intelligence through user-generated content, enable collaborative work, and deliver rich user experiences via desktop-like interfaces (Greaves, 2007; O'Reilly, 2005). Today, Web 2.0 has made its way into knowledge management as well as into technology-enhanced learning.

SOP supports experience management, requirements engineering

Therefore, it has been decided to develop the *Software Organization Platform* (SOP) based on Web 2.0 tools. SOP 1.0 was developed at Fraunhofer IESE initially by Rech, Decker, and Ras and is based on the basic principles and concepts of the RIKI system of the project RISE (Rech et al., 2007b). SOP intends to support specific software engineering activities such as experience management, process modeling, requirements engineering, and project management (Decker, Ras, Rech, Jaubert, & Rieth, 2007; Ras, Carbon, Decker, & Rech, 2007). The main motivation of an SOP is to provide integrated access to information, experiences, and learning content.

Structure of this section

The learning space approach has been implemented as a plugin in SOP (see Section 6.2), including tools for describing and annotating experience packages and learning content (see Section 6.3). This supports the conceptual as well as technical integration of knowledge/experience management and technology-enhanced learning. In addition, open source tools that have been used to develop the domain ontology and the instructional design models are named.

6.1 Research Objective and Requirements

Objective 3

Objective 3: Develop a tool for the systematic, context-aware adaptation and presentation of learning spaces based on the conceptual models.

- Develop a lightweight experience management system
- Develop a tool for the easy creation and annotation of instructional and situational content elements as well as experience packages

- Develop algorithms for adapting and generating learning spaces

6.2 Realization in the Software Organization Platform

Advantages of Wikis

A Wiki system, by definition, is "the simplest online database that could possibly work" (Leuf & Cunningham, 2001). Wikis have been used as platforms for documentation, minutes, glossaries, or repositories for additional learning materials. Their advantages are fast installation, easy adaptation to educational purposes, no acquisition costs, and intuitive usage. In the RISE project, Wikis proved that they are a good facility for *one place publishing,* meaning that there is only one version of a document available that is regarded as the current version; *simple and safe collaboration,* which refers to versioning and locking mechanisms that most Wikis provide; *easy linking,* meaning that documents within a Wiki can be linked by their title using a simple markup; *description on demand,* which means that links can be defined to pages that have not been created yet, but might be filled with content in the future (Decker, Rech, Ras, Klein, & Hoecht, 2006).

Plugins developed for this work

Therefore, SOP 1.0 consists of a MediaWiki application with the Semantic MediaWiki extension installed, and a set of plugins (as depicted in Figure 32). The following plugins have been developed for this work (white areas in Figure 32):

- Experience management plugin – serves to create, edit, and annotate new experience packages
- Learning element authoring plugin – serves to create or extract learning content from the Wiki, edit, and annotate learning elements; a vocabulary editor is used to define the metadata set for learning elements
- Learning space generation plugin – serves to adapt and present learning spaces in SOP

SOP supports the experience factory and the project organization

It can be seen that SOP supports both the experience factory and the project organization. The Wiki is used to document software engineering artifacts, classify them, and build relationships between them. These structures realize the domain model described in Section 4.2.3. Three main plugins developed in this work support the creation of experience packages and learning content on the one hand and the automatic, context-aware generation of learning spaces on the other hand. The adaptive navigation techniques of direct guidance, link sorting, link hiding, link generation, and map adaptation have been implemented in addition to the adaptive presentation techniques conditional text, page variants, fragment variants, and frame-based technique.

An ontology was developed to realize the domain model

Ontologies are becoming a widely used tool for modeling knowledge in adaptive web systems (see, for example, (Chen & Mizoguchi, 2004; Denaux, Dimitrova, & Aroyo, 2004; N. Henze, 2005). The languages RDF (Manola & Miller, 2004), RDF-Schema (Brickley & Guha, 2004), and OWL (Smith, Welty, & McGuinness, 2004) define language constructs that can be used to define ontologies in a way suitable for machine reasoning. Therefore, the domain model was realized by means of a refined ontology in the OWL-DL format by using the software engineering Body of Knowledge ontology (*Guide to the Software Engineering Body of Knowledge, SWEBOK*, 2004) and the classification of the Association for Computing Machinery (Consortium, 2005) as a baseline, which has been extended. The open source ontology editor Protégé was used for developing ontologies and exporting the OWL-DL files. The application programming interface RAP-RDF API was used for the building, storage, and retrieval of the RDF models, and the RDF query language SPARQL was used as the query language for the OWL files.

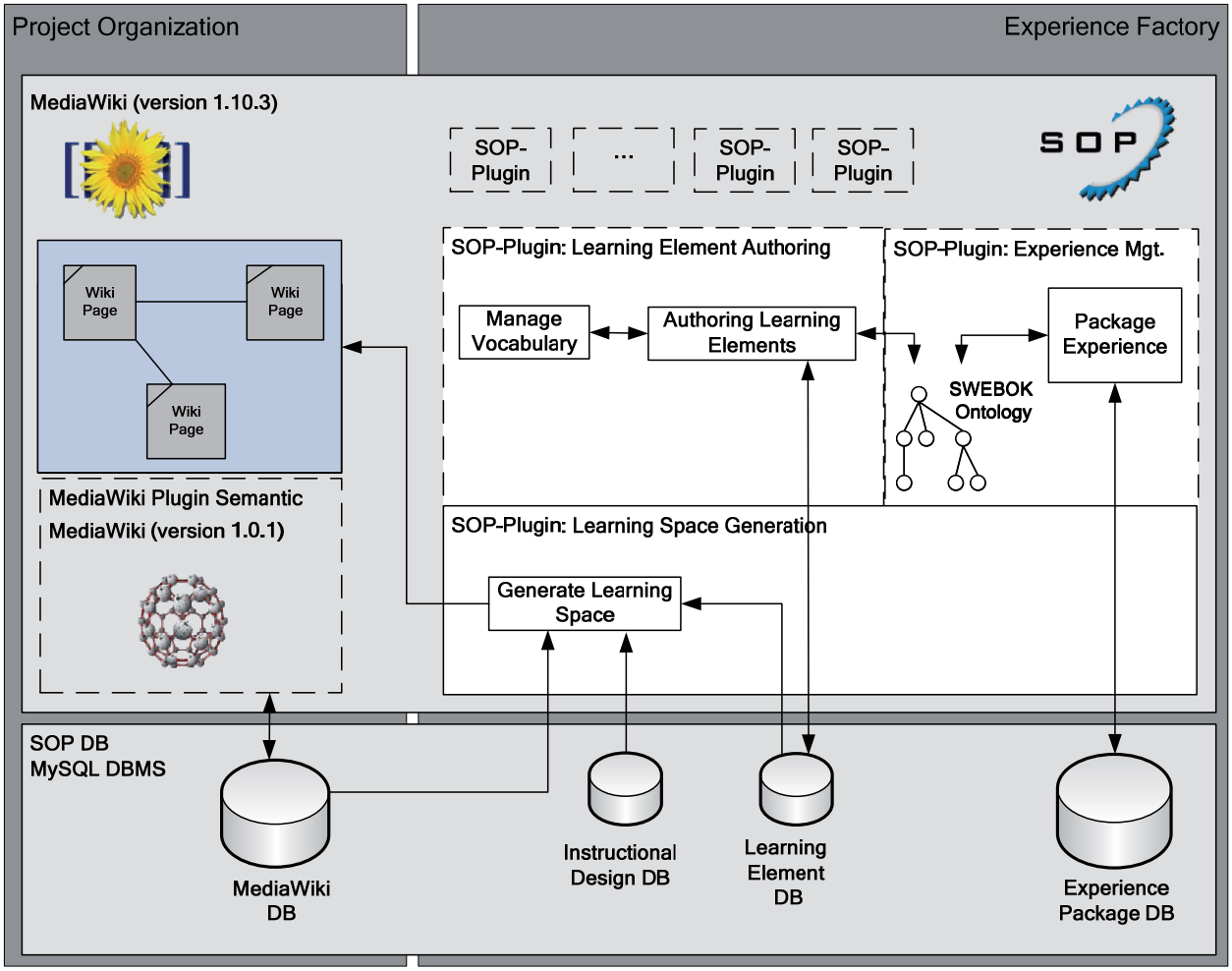


Figure 32 Schematic overview of SOP and learning space approach

Several models were developed in XML

The learning space structure template, the learning goal structure template, the decision model, the user model (as part of the context model), and the resolve model are stored in XML and are conformant to corre-

sponding XML schemas. The *Reload* editor was used to build a learning space structure template conformant to the IMS Learning Design.

The whole learning generation approach was implemented using the object-oriented programming language PHP 5. All the databases together form the SOP database, which contains the information of the experience factory as well the project organization's information documented in the Wiki. MySQL was used as relational database management system for all the plugins.

6.3 Frontend of Learning Space Approach

The Wiki technology promises a lightweight solution for capturing, organizing, and distributing emergent knowledge, and serves as a basis for structuring and presenting learning spaces. The following sections provide short descriptions and screen shots of the user frontend when using Wiki technology.

6.3.1 Situational Content and Experience Package

Situational content elements build the context model

SOP harnesses collaborative generation (i.e., quick and easy page creation and linkage) and semantic annotations (e.g., tagging) of content via the Wiki for the software engineering artifacts *products*, *processes*, *projects*, *individuals*, *groups*, *customers*, *organizations*, and *software tools*. These situational content elements form the context model of the learning space approach.

Wiki categories and semantic relationships

In order to classify these core Wiki pages, Wiki categories (syntax: `[[Category:categoryName]]`) are used. They classify the Wiki pages into multiple, freely named categories. In addition, by using the features of the Semantic MediaWiki, specific semantic relationships (syntax: `[[relationshipName::wikiPageName]]`) can be defined between instances of the Wiki pages and categories. These categories are predefined by the context model – the instances (i.e., concrete Wiki pages) are referenced when an experience package's context is described.

SOP offers functionality for creating, editing, and deleting new experience packages. The edit function also provides the functionality to annotate the experience package by relating it to existing Wiki pages in SOP in order to describe its context.

Figure 33 shows an example of an experience package in SOP. A tab in the top left corner can be used to generate a learning space for the experience package shown (see Figure 15 for an example of a learning space).

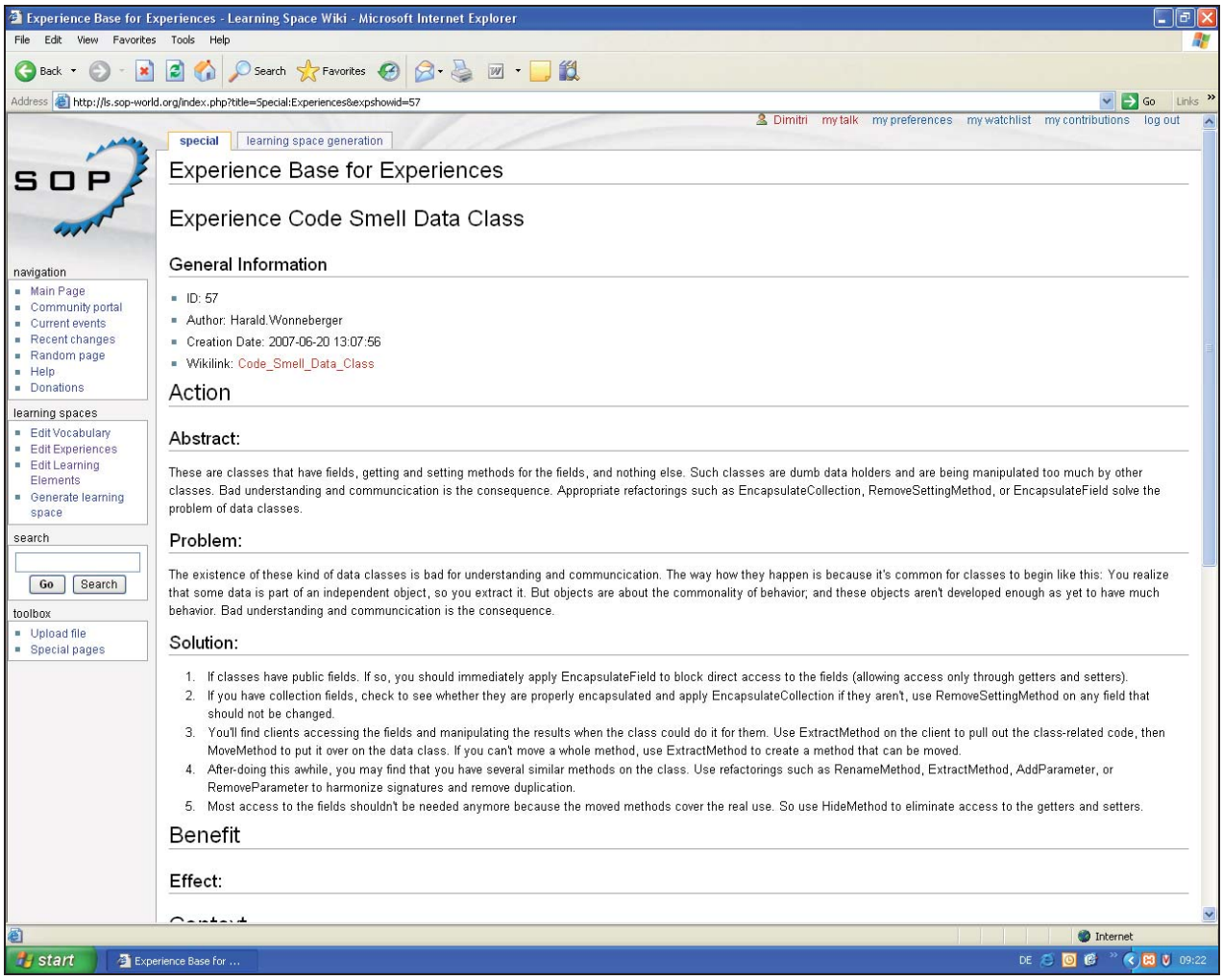


Figure 33 Experience package

6.3.2 Instructional Content

Wiki pages can be transformed into learning elements

Besides the Wiki pages of the specific software engineering categories, other kinds of software engineering information may be described, e.g., definitions, explanations, conclusions, etc. The knowledge engineer can easily transform Wiki pages into learning elements. The requirement of easy annotation of learning elements is fulfilled by a set of pre-defined values and metadata attributes for classifying learning elements being offered. This metadata set is defined in SOP by using the *Vocabulary Manager* (see Section 6.3).

Vocabulary manager

A so-called vocabulary manager allows creating, editing, and deleting metadata attributes as well as related values (e.g., attribute: illustration; values: example, counter-example), i.e., it is used by the knowledge engineer to develop the learning element taxonomy.

Annotate learning elements by using keywords

In addition to the classification of learning elements, keywords can be used to annotate the learning elements. These keywords are retrieved from the software engineering domain ontology.

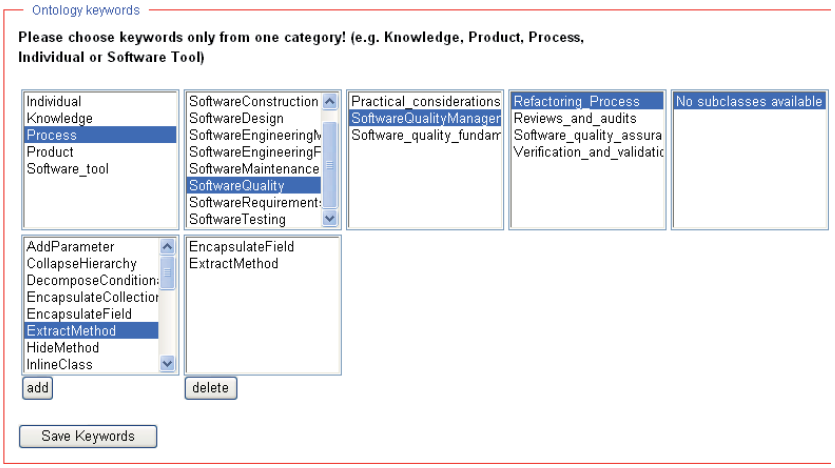


Figure 34 Annotation of learning elements

Figure 35 illustrates the main entry of the learning element authoring plugin.

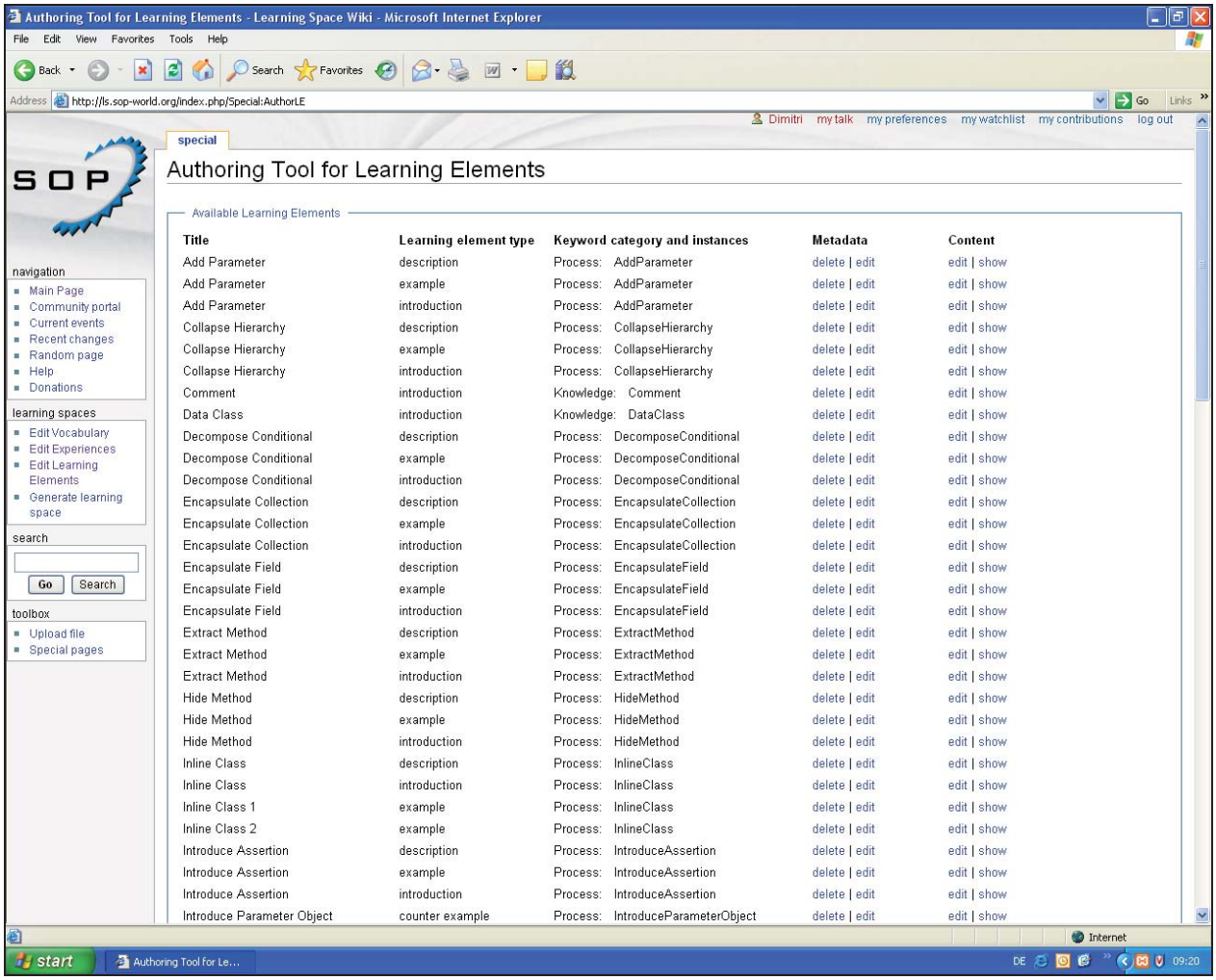


Figure 35 Authoring tool for learning elements

6.3.3 Learning Space

When the software developer or the project manager decides to access a learning space, he clicks on the button above the experience package description Figure 33. After selecting an overall learning goal, an overview of the learning space is shown (see Figure 36). In this example, the user has chosen the overall learning goal level of "remember". One instance of the domain concept *knowledge* and seven instances of the domain concept *process* are listed.

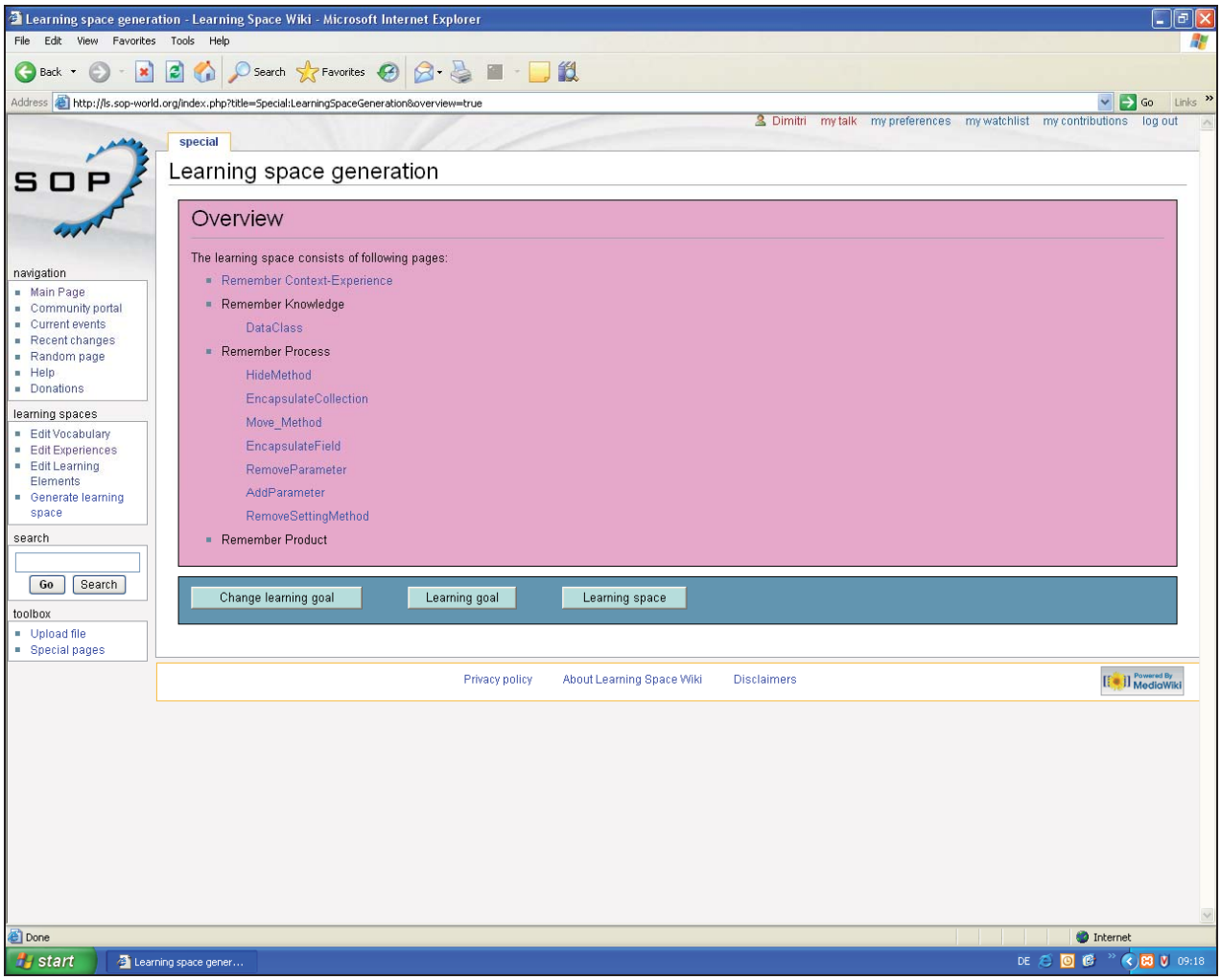


Figure 36 Overview of a learning space on the "remember" learning goal level

7 Empirical Evaluation – A Controlled Experiment

"Experimentation in software engineering is necessary but difficult. Common wisdom, intuition, speculation, and proofs of concept are not reliable sources of credible knowledge" (V. R. Basili)

Goals of a learning space

The goal of a learning space is to improve the understanding of the information provided by an experience package. Better understanding of the experience package is expected to lead to more efficient application of the experience package to the current working situation. The application of the experience creates new factual, conceptual, and procedural knowledge, which is the basis for new skills and competence development (i.e., knowledge acquisition). Further, the improvement of the perceived information quality was assessed.

Results

The statistical data analysis revealed that all eleven null hypotheses related to understanding (i.e., understanding correctness), knowledge acquisition variables (i.e., in general and for five different cognitive levels), application (i.e., efficiency, completeness, accuracy), and perceived information quality could be rejected, which means that learning spaces have a high potential for improving experience package reuse. Learning spaces provide significantly higher:

- **understanding correctness** ($p = .002$, 21% improvement < 25%)
- **knowledge acquisition** in general ($p = .000$, 219% improvement > 50%)
  - knowledge acquisition at the level of ***remembering*** ($p = .001$, 230% improvement > 50%)
  - knowledge acquisition at the level of ***understanding*** ($p = .001$, 275% improvement > 50%)
  - knowledge acquisition at the level of ***applying*** ($p = .006$, 81% improvement > 50%)
  - knowledge acquisition at the level of ***analyzing*** ($p = .004$, 198% improvement > 50%)
  - knowledge acquisition at the level of ***creating*** ($p = .029$, 121% improvement > 50%)
- **application efficiency** ($p = .006$, 53% improvement > 25%)
- **application completeness** ($p = .006$, 51% improvement > 25%)

- **application accuracy** (p = .049, 28% improvement > 25%)
- **perceived information quality** (p = .013, 18% improvement < 25%)

The related measures of these metrics can be found in Section 7.1.2.2.

The experiment is an exploratory evaluation that builds a strong baseline for future research

In order to investigate the effect of learning spaces on experience reuse, a counterbalanced, within-subject, two-factorial experiment was conducted with 19 undergraduate and graduate students of the University of Kaiserslautern. Due to the fact that experiments related to the didactical enrichment of software engineering experience packages have not been conducted before, this experiment serves as an exploratory evaluation that can be used as a baseline for future evaluations and developments in this area. Therefore, a strong emphasis has been put upon the construction of reliable measurement instruments, the selection of suitable disturbing factors for controlling the experiment, and upon the data analysis.

In this chapter, the experiment planning includes the detailed research hypotheses, the experimental variables, the experimental design, as well as the experiment execution and the procedure for data analysis. Afterwards, the execution of a principal component analysis will be described for the disturbing factors related to the experience levels of the subjects in order to increase the reliability of these scales. A detailed item analysis, which is usually applied in educational test construction, was applied for selecting suitable test items for the questionnaire related to knowledge acquisition. The central part of this chapter is the description of the experiment's results. Particular emphasis is placed on the analysis of confounding effects. A literature survey was necessary to clarify the terminology and identify the different statistical approaches for the identification and the correction confounding effects. The reason for this survey was that these effects are either mostly neglected in empirical software engineering or examined incorrectly. The hypothesis tests were only performed when their assumptions were fulfilled. Especially for the usage of the analysis of covariance, several detailed tests were compulsory. A post-hoc power analysis and threats to validity conclude this chapter.

7.1 Evaluation Goal and Experiment Planning

Figure 37 shows the experimental model and the metrics used to investigate understanding, knowledge acquisition, application, and perceived information quality (in italics).

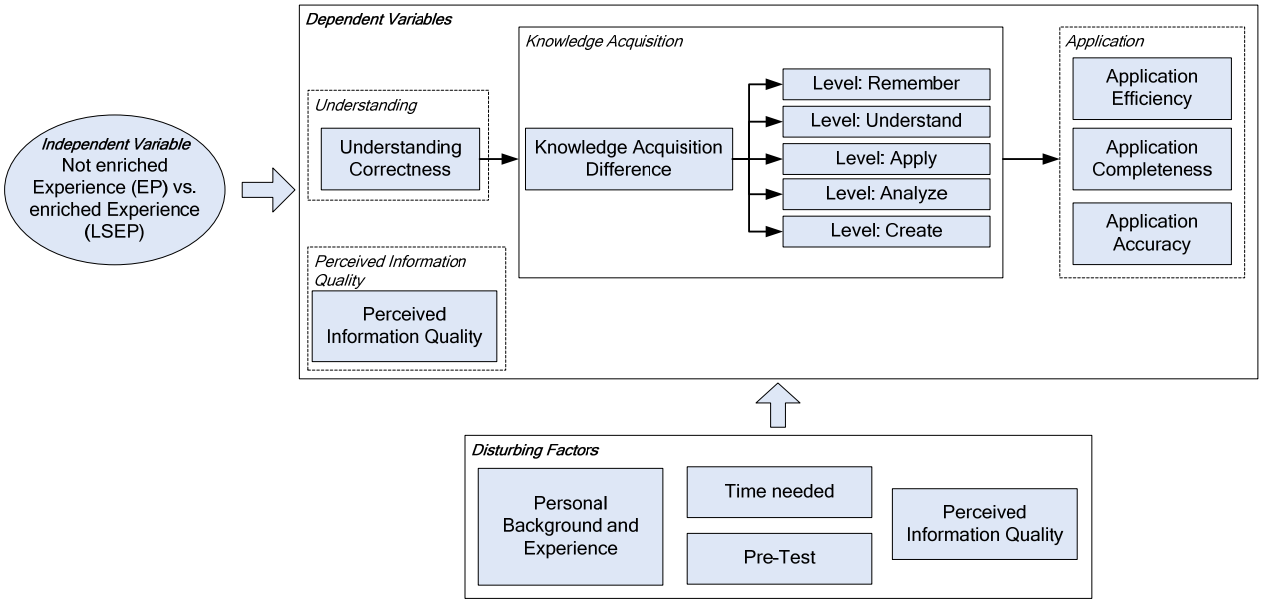


Figure 37 Experimental model

GQM goal

Hence, in accordance with the GQM approach the research goal can be stated (V. R. Basili, Caldiera, Rombach et al., 2002) as:

Analyze *the effect of learning spaces on experience package reuse* for the purpose of *evaluation* with respect to

- *understanding correctness,*
- *knowledge acquisition differences in total and on the cognitive levels of remembering, understanding, applying, analyzing, and creating*
- *application efficiency, completeness, and accuracy,*
- *perceived information quality*

from the viewpoint of the *researcher* in the context of *a controlled experiment in the domain of experience package reuse at the University of Kaiserslautern*.

All the dependent variables can only be measured indirectly:

Measuring *understanding correctness*

Understanding correctness was measured by the total score of correctly answered questions related to the experience package, which were part of the post-test questionnaire (see Appendix A.2). The questions were weighted according to their difficulty by means of assigning different numbers of points.

Metrics related to *knowledge acquisition* were measured based on the correctly answered questions with different numbers of points according to their difficulty (see Appendix A.2).

Knowledge acquisition difference (in total)

Knowledge acquisition difference was calculated as the difference between the score of a pre-test and the score of a post-test. The pre-test consists of a set of weighted questions (see Appendix A.2), which were answered by the students before the experimental run. The post-test is composed of the same questions as the pre-test, but was filled out after the experimental run. These questions of the pre-test and post-test make use of new application examples, i.e., new situations (e.g., unknown software code) in order to investigate the ability to transfer newly acquired knowledge to unknown problems.

Knowledge acquisition difference on cognitive levels

Knowledge acquisition difference regarding the different *cognitive levels* was also calculated based on the difference between the scores of the pre-test and the post-test. But only questions assigned to the same cognitive level (i.e., remember, understand, apply, analyze, and create) were considered for each of the five levels.

Metrics related to *application* were measured by analyzing the outcome of practical assignments in the domain of refactoring (see an example of such an assignment in Appendix A.4). The assignments were based on two main tasks: first, identifying and marking code smells in code fragments on paper and second, describing how they can be removed and writing down the result after refactoring. Scores for both tasks were assigned before the experiment was performed by two refactoring experts according to the difficulty of the code smells' identification and removal.

Application efficiency

Application efficiency was determined by means of dividing the total score for identification and removal by the time spent.

Application completeness

Application completeness was measured by means of dividing the total score for identification and removal of code smells by the highest score possible for all assignments in the experimental run.

Application accuracy

Application accuracy focuses on the identification of code smells. It was measured by means of the number of correct defects found in the code divided by the defects indicated as a defect by the subjects.

Perceived information quality

Perceived information quality was captured by specific questions in the debriefing questionnaire (see Appendix A.5). The questions asked the subject how useful, boring, easy, clear, and complete the provided information in an experience package, respectively learning space, was.

A more formal definition of the data collection of these dependent variables can be found in Section 7.1.2.2 in Table 42.

The experiment relies only on two independent factors in order to find significant results

The experiment was conducted with 19 undergraduate as well as graduate computer science students of the University of Kaiserslautern. The problem with experiments in software engineering is that they have to be executed with a small number of subjects, compared, for example, to empirical investigations in social science where the number of subjects is higher and the results are more representative of the whole population. In addition, software engineering is a young field of research, which has only existed for about 40 years. Hence, in many software engineering fields, significant results from empirical research are either just not available or experiments have focused on many factors with a small number of subjects – further replications would be necessary in order to produce significant findings with acceptable power. Therefore, this experiment relies on only two independent factors with only two alternatives each and tries to find significant results based on a small number of students (<20). A detailed experiment documentation and analysis ensures that the experiment can be easily replicated in the future to confirm the results of this experiment. Before the hypotheses are stated in the next section, the different variables and their formal notations are introduced.

Table 41 Notations used in the controlled experiment

Term	Definition
LSEP	Experience package enriched with a learning space
EP	Conventional experience package with no learning space augmentation
Independent Variables	
treatment	- Experience package enriched with a learning space (EP) - Conventional experience package with no learning space enrichment (LSEP)
sequence	- sequence LSEP | EP - sequence EP | LSEP
Dependent Variables	
$ucorr_i$ (EP, LSEP)	The individual *understanding correctness* for subject i when using (EP, LSEP)
$know\_diff_i$ (EP, LSEP)	The individual *knowlegde acquisition difference* for subject i when using (EP, LSEP)
know_diff_x (EP, LSEP)	The individual knowlegde *acquisition difference* for subject i for the cognitive dimension (*remember, understand, apply, analyze, create*) when using (EP, LSEP)
$aeff_i$ (EP, LSEP)	The individual *application efficiency* for subject i when using (EP, LSEP)
$acomp_i$ (EP, LSEP)	The individual *application completeness* for subject i when using (EP, LSEP)
$aaccu_i$ (EP, LSEP)	The individual *application accuracy* for subject i when using (EP, LSEP)
inf_qua_i (EP, LSEP)	The individual *perceived information quality* for subject i when using (EP, LSEP)

Disturbing Factors	
exp_dev_i	The individual software development experience of subject i
exp_jp_i	The individual Java programming experience of subject i
exp_ref_i	The individual refactoring experience of subject i
exp_sqa_i	The individual quality assurance experience of subject i
exp_main_i	The individual software maintenance experience of subject i
tn_i	The individual time needed of subject i
$pre\text{-}test_i$	The individual pre-test score of subject i
$inf\_qua\_LSEP_i$	The individual perceived information quality (LSEP) of subject i
$inf\_qua\_EP_i$	The individual perceived information quality (EP) of subject i

7.1.1 Detailed Statistical Hypotheses

Statistical alternative hypotheses

For the experiment, alternative statistical hypotheses were derived from the first four hypotheses stated in Section 1.4.

$H_{1.1}$ "Average understanding correctness" – The average understanding correctness *ucorr* of the experimental group (related to the usage of (LSEP)) is higher than the average understanding correctness *ucorr* of the control group (related to the usage of (EP)).

$$\mu(ucorr_{LSEP}) > \mu(ucorr_{EP})$$

$H_{1.2.1}$ "Average knowledge acquisition difference" – The average knowledge acquisition difference of the experimental group (related to the usage of (LSEP)) is higher than the average knowledge acquisition difference of the control group (related to the usage of (EP)).

$$\mu(know\_diff_{LSEP}) > \mu(know\_diff_{EP})$$

Hypothesis $H_{1.2.1}$ has been refined further with regard to the different cognitive processes:

$H_{1.2.2.x}$ "Average overall knowledge acquisition difference related to the cognitive processes" – The average knowledge acquisition difference of the experimental group (related to the usage of (LSEP)) is higher than the average knowledge acquisition difference of the control group (related to the usage of (EP)) for the cognitive processes X (*remember*, *understand*, *apply, analyze*, and *create)*:

$$\mu(know\_diff\_x_{LSEP}) > \mu(know\_diff\_x_{EP})$$

$H_{1.3.1}$ "Average application efficiency" – The average application efficiency *aeff* of the experimental group (related to the usage of (LSEP)) is higher than the average application efficiency *aeff* of the control group (related to the usage of (EP)).

$$\mu(aeff_{LSEP}) > \mu(aeff_{EP})$$

$H_{1.3.2}$ "Average application completeness" – The average application completeness *acomp* of the experimental group (related to the usage of (LSEP)) is higher than the average application completeness *acomp* of the control group (related to the usage of (EP)).

$$\mu(acomp_{LSEP}) > \mu(acomp_{EP})$$

$H_{1.3.3}$ "Average application accuracy" – The average application accuracy *aaccu* of the experimental group (related to the usage of (LSEP)) is higher than the average application accuracy *aaccu* of the control group (related to the usage of (EP)).

$$\mu(aaccu_{LSEP}) > \mu(aaccu_{EP})$$

$H_{1.4}$ "Average information quality" – The average perceived information quality *inf_qua_LSEP* of the experimental group (related to the usage of (LSEP)) is higher than the average perceived information quality *inf_qua_EP* of the control group (related to the usage of (EP)).

$$\mu(\inf\_qua_{LSEP}) > \mu(\inf\_qua_{EP})$$

Statistical null hypotheses

The alternative hypotheses express that learning spaces lead to higher competence development and a higher number of well understood and hence correctly applied experience packages, while the quality of the provided information increases. The following null hypotheses have to be tested:

$H_{0.1}$ "Average understanding correctness" – The average understanding correctness *ucorr* of the experimental group (related to the usage of (LSEP)) is equal to or lower than the average understanding correctness *ucorr* of the control group (related to the usage of (EP)).

$$\mu(ucorr_{LSEP}) \leq \mu(ucorr_{EP})$$

$H_{0.2.1}$ "Average knowledge acquisition difference" – The average knowledge acquisition difference of the experimental group (re-

lated to the usage of (LSEP)) is equal to or lower than the average knowledge acquisition difference of the control group (related to the usage of (EP)).

$$\mu(know\_diff_{LSEP}) \leq \mu(knowl\_diff_{EP})$$

$H_{0.2.2.X}$ "Average knowledge acquisition difference related to the cognitive knowledge dimension" – The average knowledge acquisition difference of the experimental group (related to the usage of (LSEP)) is equal to or lower than the average knowledge acquisition difference of the control group (related to the usage of (EP)) for the cognitive process dimensions X *remembering*, *understanding*, *applying*, *analyzing*, and *creating*.

$$\mu(know\_diff\_x_{LSEP}) \leq \mu(know\_diff\_x_{EP})$$

$H_{0.3.1}$ "Average application efficiency" – The average application efficiency *aeff* of the experimental group (related to the usage of (LSEP)) is equal to or lower than the average application efficiency *aeff* of the control group (related to the usage of (EP)).

$$\mu(aeff_{LSEP}) \leq \mu(aeff_{EP})$$

$H_{0.3.2}$ "Average application completeness" – The average application completeness *acomp* of the experimental group (related to the usage of (LSEP)) is equal to or lower than the average application completeness *acomp* of the control group (related to the usage of (EP)).

$$\mu(acomp_{LSEP}) \leq \mu(acomp_{EP})$$

$H_{0.3.3}$ "Average application accuracy" – The average application correctness *aaccu* of the experimental group (related to the usage of (LSEP)) is equal to or lower than the average application correctness *aaccu* of the control group (related to the usage of (EP)).

$$\mu(aaccu_{LSEP}) \leq \mu(aaccu_{EP})$$

$H_{0.4}$ "Average perceived information quality" – The average perceived information quality *inf_qua* of a learning space is higher than the average perceived information quality *inf_qua* of an experience package.

$$\mu(\inf\_qua_{LSEP}) \leq \mu(\inf\_qua_{EP})$$

7.1.2 Experimental Variables

During the experiment, three types of variables are under observation: independent variables (i.e., factors), dependent variables (i.e., response variables), and disturbing factors (i.e., undesired variables).

7.1.2.1 Independent Variables

Treatment

Two independent variables exist. The first independent variable *treatment* is related to the information provided to the subject during the reuse of an experience package. Two alternatives are used during the experiment. Alternative 1, i.e., (EP), refers to the usage of standard experience package descriptions used in the past. Alternative 2, i.e., (LSEP), refers to providing the subject with augmented experience packages, i.e., learning spaces enriching an experience package. The second variable *sequence* is used for the investigation of sequence effects, i.e., whether the order of treatments has an impact on the treatment effect (see Section 7.4.6 and Section C.5.2 for further details).

Sequence

7.1.2.2 Data Collection of Dependent Variables

In the following, the measurement of the dependent variables is made explicit by providing formulas and descriptions of the direct measures used. The detailed questionnaires can be found in the Appendix.

Table 42 Data collection of dependent variables

Depemdemt Variable	Name	Formula	Range	Direct Measures	Description
ucorr	Understanding correctness	$\sum_{i=1}^{\#ucorr\_questions} ucorr\_question_i$ Sum of all questions asking the subject facts and relationships about the experience package and its related context. For each question, a maximum number of points can be obtained when it is answered correctly.	$R = [0, score\_max]$	ucorr_question$_i$	The answers to specific questions are evaluated regarding their correctness. They are part of the post-test questionnaire and have either a binary scale with correct answers encoded as "1" and incorrect answers encoded as "0", or a numerical scale that depends on the difficulty of the question.
know_diff	Knowledge acquisition difference	$\sum_{i=1}^{\#questions} postt\_question_i - \sum_{i=1}^{\#questions} pret\_question_i$ Difference between the sum of all post-test questions asking the subject facts and relationships about the information provided in an experience package, respectively learning space, minus the sum of all pre-test questions. For each question a maximum number of points can be obtained when it is answered correctly. Questions address all the cognitive levels of remembering, understanding, applying, analyzing, and creating.	$R = [0, score\_max]$	postt_question$_i$	The answers of the entire knowledge questionnaire are evaluated regarding their correctness. They have a binary scale with correct answers encoded as "1" and incorrect answers encoded as "0", or a numerical scale that depends on the difficulty of the questions. The test is performed after the experiment treatment.
				pret_question$_i$	The test consists of the same questions as the post-test and is performed before the experiment treatment.
know_diff_x	Knowledge acquisition diff. on cog. level	$\sum_{i=1}^{\#questions\_x} postt\_question\_x_i - \sum_{i=1}^{\#questions\_x} pret\_question\_x_i$	$R = [0, score\_max]$	postt_question_x$_i$	The answers of the questions for a cognitive level x are evaluated regarding their correctness. They have a binary scale with correct answers encoded as "1" and incorrect answers encoded as "0", or a numerical scale that depends on the difficulty of the questions. The test is performed after the experiment treatment.

				pret_question_x_i	The test consists of the same questions as the post-test and is performed before the experiment treatment.
aeff	Application efficiency	$\frac{\sum_{i}^{\#defects} score\_ident_i + \sum_{i}^{\#defects} score\_remove_i}{application\_time}$ Division of the total score for identifying and removing defects by the total time spent on refactoring.	$R = \Re$	score_ident	Each subject has to identify and mark defects (i.e., code smells) in assignments. Each assigment is related to an experience package, respectively learning space. Score_ident consists of points received for correctly identifying and marking a specific defect.
				score_remove	Each subject has to remove the identified defects in the assignments. Score_remove consists of points received for correctly removing a detected defect.
				applica-tion_time	Time used for the identification and removal of a defect.
acomp	Application completeness	$\frac{\sum_{i}^{\#defects} score\_ident_i + \sum_{i}^{\#defects} score\_remove_i}{\max\_score}$ Division of the total score for identifying and removing defects by the highest score possible for all assigments in the experimental run.	$R = [0,1]$	score_ident, score_remove	See above
				max_score	The maximum score of 12.
aaccu	Application accuracy	$\frac{\#correct\_defects\_found}{\#defects\_indicated\_as\_defect}$ Division of the total number of correct defects found in the code by the defects indicated as a defect by the subjects.	$R = [0,1]$	#cor-rect_defects_found	Correct_defects_found consists of the number of correct defects found by a subject.
				#de-fects_indicated_as_defect	This measure corresponds to the total number of defects identified as a defect by the subject.
inf_qua	Perceived information quality	$\frac{\sum_{i=1}^{\#\inf\_qua\_questions} question\_\inf\_qua\_score_i}{\#\inf\_qua\_questions}$	$R = [1,7]$	ques-tion_inf_qua_score$_i$	Each question uses a seven-point Likert-type scale (Likert, 1932). It reflects the subjective perception of the subjects about the quality of the information provided in an experience package, respectively a learning space. Five possible dimensions were assessed (“useful” vs. “useless”, “absorbing” vs. “boring”, “easy” vs. “difficult”, “clear” vs. “confusing”, “complete vs. incomplete”.)
				#inf_qua_ques tions	#inf_qua_questions = 5

7.1.2.3 Data Collection of Disturbing Factors

The values of the undesired variables are measured by questionnaires that all subjects must complete (see Table 37). Part of the questions were filled in before the experiment started (i.e., *briefing* questionnaire), the other questions after the experimental periods (i.e., *debriefing* questionnaire). The following table provides a more detailed description of the disturbing factors.

7.1.3 Experiment Description

7.1.3.1 Subjects

Subjects: undergraduate and graduate students

The experimental subjects were undergraduate and graduate students with an average study time in computer science of 3.5 years and an average software development experience of 4.2 years. The students of the University of Kaiserslautern were enrolled in a three-semester class about software engineering fundamentals and special software engineering courses. Some of them had attended specific SE courses (e.g., quality assurance, process modeling, or project management). The courses were supplemented by practical sessions. All subjects took part in a practicum, where a software system was developed during a 13-week period. One requirement of the experiment's design was that the experiment must be perfectly integrated into the system's development process. This ensured that the students were motivated.

Table 43 Data collection of disturbing factors

Response Variable	Name	Formula	Range	Direct Measures	Description
exp_dev	Experience in software development	$\frac{\sum_{i=1}^{\#\text{exp\_dev\_questions}} question\_\text{exp}\_dev\_score_i}{\#\text{exp}\_dev\_questions}$	$R=[0,6]$	question_exp_dev_score$_i$	The questions of this category measure the experience level in *software development* in general (See Appendix A.1). The questions use a 7-point Likert-type scale (Likert, 1932). Each answer was mapped to the value range of R = [0,6] assuming equal distances between possible answers ("no experience" is encoded as "0", "very low experience" is encoded as "1", "low experience" as "2", "neutral" is encoded as "3", "medium experience" is encoded as "4", "high experience" is encoded as "5",and "expert" is encoded as "6").
				#exp_dev_questions	An experience questionnaire gathers the experience of the subjects. Dividing the total score by the number of questions provides a normalized value range of R = [0,6].
exp_jp	Experience in java programming	$\frac{\sum_{i=1}^{\#\text{exp\_jp\_questions}} question\_\text{exp}\_jp\_score_i}{\#\text{exp}\_jp\_questions}$	$R=[0,6]$	question_exp_jp_score$_i$	The questions of this category measure the experience level in *Java programming*. The questions are coded the same as exp_dev.
				#exp_jp_questions	Dividing the total score by the number of questions provides a normalized value range of R = [0,6].
exp_ref	Experience in refactoring	$\frac{\sum_{i=1}^{\#\text{exp\_ref\_questions}} question\_\text{exp}\_ref\_score_i}{\#\text{exp}\_ref\_questions}$	$R=[0,6]$	question_exp_ref_score$_i$	The questions of this category measure the experience level in *refactoring*. The questions are coded the same as exp_dev.
				#exp_ref_questions	Dividing the total score by the number of questions provides a normalized value range of R = [0,6].
exp_sqa	ence in software quality assur-	$\frac{\sum_{i=1}^{\#\text{exp\_sqa\_questions}} question\_\text{exp}\_sqa\_score_i}{\#\text{exp}\_sqa\_questions}$	$R=[0,6]$	question_exp_sqa_score$_i$	The questions of this category measure the experience level in *software quality assurance*. The questions are coded the same as exp_dev.
pre-test	Pre-test score	$\sum_{i=1}^{\#questions} question\_pret_i$	$R=[0,\text{ma}$	question_pret$_i$	The answers of the complete knowledge pre-test questionnaire from the first day are evaluated in terms of their correctness. This measure corresponds to the total score obtained by a subject.

Response Variable	Name	Formula	Range	Direct Measures	Description
				#exp_sqa_questions	Dividing the total score by the number of questions provides a normalized value range of R = [0,6].
exp_main	Experience in software maintenace	$\frac{\sum_{i=1}^{\#\text{exp\_main\_questions}} question\_\text{exp}\_main\_score_i}{\#\text{exp}\_main\_questions}$	$R=[0,6]$	question_exp_main_score$_i$	The questions of this category measure the experience level in *software maintenance*. The questions are coded the same as exp_dev.
				#exp_main_questions	Dividing the total score by the number of questions provides a normalized value range of R = [0,6].
tn	time needed	$\frac{\sum_{i=1}^{\#tn\_questions} question\_tn_i}{\#tn\_questions}$	$R=[0,1]$	question_tn$_i$	Five questions were posed about the *Time Needed*. If a subject had the feeling that there was not enough time to fulfill a specific task, then "yes" (encoded as "1") is marked, otherwise "no" (encoded as "0").
				#tn_questions	Simply adding the scores per answer provides a measure of time needed with a minimal score of 0 and a maximum score of 5. Dividing the number of questions provides a normalized value range of R = [0,1].
inf_qua_LSEP	Perceived information quality of learning space	$\frac{\sum_{i=1}^{\#\text{inf}\_qua\_LSEP\_questions} question\_\text{inf}\_qua\_LSEP\_score_i}{\#\text{inf}\_qua\_LSEP\_questions}$	$R=[1,7]$	question_inf_qua_LSEP_score$_i$	Each question uses a seven-point Likert-type scale (Likert, 1932). It reflects the subjective perception of the subjects about the quality of the information provided in a learning space. Five dimensions were possible ("useful" vs. "useless", "absorbing" vs. "boring", "easy" vs. "difficult", "clear" vs. "confusing", "complete vs. incomplete").
				#inf_qua_LSEP_questions	Dividing the total score by the number of questions provides a normalized value range of R = [1,7].
inf_qua_EP	Information quality of experience package	$\frac{\sum_{i=1}^{\#\text{inf}\_qua\_EP\_questions} question\_\text{inf}\_qua\_EP\_score_i}{\#\text{inf}\_qua\_EP\_questions}$	$R=[1,7]$	question_inf_qua_EP_score$_i$	Each question uses a seven-point Likert-type scale (Likert, 1932). It reflects the subjective perception of the subjects about the quality of the information provided in an experience package. Five dimensions were possible ("useful" vs. "useless", "absorbing" vs. "boring", "easy" vs. "difficult", "clear" vs. "confusing", "complete vs. incomplete").
				#inf_qua_EP_questions	Dividing the total score by the number of questions provides a normalized value range of R = [1,7].

7.1.3.2 Experimental Design

Design: counter-balanced within-subject two-factorial experiment

For evaluating the effect of learning spaces, a counterbalanced, within-subject, two-factorial design was selected. An experiment is balanced when all alternatives of the independent variable (or treatment groups) have the same number of experimental units. More details about this design can be found in (Winer, Brown, & Michels, 1991). One independent variable (i.e., factor) is the type of information provided (i.e., EP or LSEP) and the second one is the sequence of the treatment (i.e., LSEP → EP or EP → LSEP). The reason for selecting a within-subject design was primarily the low number of available subjects (i.e., 19) and the risk of losing power if a parallel design had been used. Higher power can be achieved because such a "crossover" design removes the intersubject variability from the comparison between treatments, and can provide unbiased estimates for the difference between treatments. In addition, the within-subject design allows reducing the error variance related to the differences amongst the subjects (e.g., related to their experience level). Half of the subjects (group 1) were assigned to use (LSEP) during the first period and the other half of the subjects (group 2) used (EP) at the same time. During the second sequence, group 1 used (EP) and group 2 used (LSEP).

To reduce period and carry-over effects (see Section 7.4.6), two different sets of experience packages were used. Otherwise, this would have invalidated the results of the second period. In order to verify the equivalence of the two sets, subjective information on the complexity of the experience packages was gathered after the experiment (i.e., by the debriefing questionnaire). The design is depicted in Figure 38.

Randomization was done regarding experience packages and the subjects' experience level

In order to prevent these undesired sources of variation from being brought into the experiment, randomization was done regarding the selection and sequence of the experience packages used and the subjects' level of experience. The assignment was to be done completely at random: Eight experience packages with two different levels of complexity were prepared for the two runs. The labels of the two experience packages with low complexity were put into one bag, the packages with high complexity were put into another bag. The sets of two experience packages were created by asking a subject to draw one label from each bag (i.e., only four experience packages were used for the experiment). The low number of subjects is the reason why even with by randomization, almost all subjects with high or low experience would be assigned to the same group. This would lead to undesired effects. Therefore, the results of the briefing questionnaire were used to assign the subjects to four groups of different experience levels (i.e., high, medium, low, or no experience). The subjects of each group were assigned randomly to group 1 and group 2. After the randomization, the questionnaires and material were selected and prepared in the right order, i.e., the right ex-

perience packages and related assignments in the right sequence. Each subject remained in the same group for both periods.

7.1.3.3 Experimental Parameters and Materials

Application domain: refactoring

The experiment was conducted in one room with the same technical equipment for each subject. For the experiment, the task of refactoring was chosen. 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 (Fowler, 1999). All participants had at least basic skills in Java programming. Hence, they knew the Java constructs and were able to understand Java code. The system used for this experiment was the complete running Java system developed by the students during their practicum. The system consists of six packages and more than 80 classes. The system has been implemented in Eclipse. All experience packages, questionnaires, and assignments were related to the topic of refactoring. The material language was English, which was no problem for the subjects.

7.1.3.4 Experimental Task and Procedure

Preparation day

The week before the experiment took place, the students attended a preparation day (see Figure 38). The goal was to inform the subjects about the experimental setting, the procedure, and the development activity (i.e., refactoring). No tool-related training was necessary, since the students had to use a normal web browser in order to access the experience packages and learning spaces in the Wiki. The experimental procedure and tasks are shown in Figure 38.

Pre- & Post-test

Both groups had to pass a pre-test and a post-test before respectively after, each experimental unit. A pre-test captures the knowledge of all the subjects before an experimental period and a post-test measures the knowledge after an experimental period.

Experiment during two consecutive days

The experiment was then conducted on two consecutive days with two experimental units per day. The subjects were only told that they are working on refactoring tasks, but they were not informed about the details of the experiment, e.g., hypotheses to be tested or the difference between the groups. During the second day, at the end of each experimental unit, a debriefing questionnaire had to be completed. The subjects returned the tests and the questionnaires back to the evaluators before they left the room.

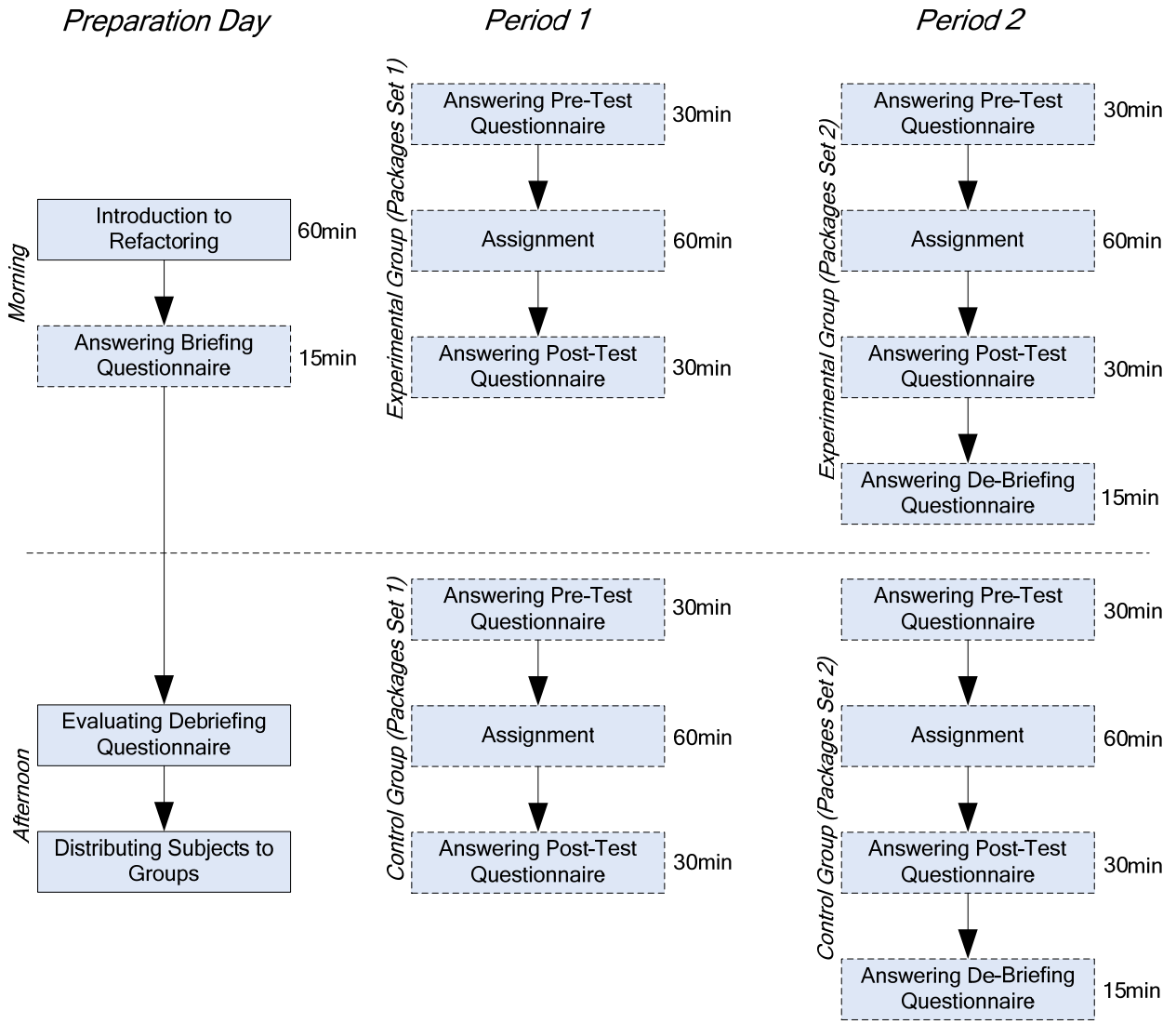


Figure 38 Experimental procedure

7.2 Data Analysis Procedure

The purpose of the statistical data analysis is to obtain precise, valid, and objective results from collected data. Not only the independent variables can have an effect on the dependent variables, but also the disturbing factors, which were also captured in the experiment. In the following, the data analysis procedure will be described for the independent and dependent variables (see Figure 39).

Principal component analysis

Item analysis

Reliability analysis

After conducting the experiment, the data was transferred from the paper-based questionnaires to SPSS (version 15) – a statistical analysis tool. First, a principal component analysis (PCA) was performed for the disturbing factors related to experience in order to construct highly reliable scales. Then, an item analysis was used to select question items for inclusion and to identify poorly written test items. The items were selected based on their discrimination index (D), discrimination coefficient I, and item difficulty (p). For these variables, reliability was determined in order to make a decision, for example, about whether the disturbing factors were suitable for reducing the measurement error in the analysis of covariance. Afterwards, an independent sample t-test was done to identify significant differences between the experiment group and the control group regarding their experience levels. An outlier and anomaly analysis was used before the first statistical tests were performed.

Normality tests

The statistical analysis requires selecting appropriate hypothesis tests. The selection depends, for example, on the number of variables to be analyzed, the purpose of the analysis, and the distribution type and scale of the variables. First, the data will be checked for normality. If the data is non-normal, then non-parametric tests have to be used, otherwise parametric tests are used. In order to find out whether the data is normally distributed, a Shapiro Wilk's W test was performed, which works fine for smaller samples.

Check for confounding effects

The dependent variables in a counterbalanced, within-subject experiment may be confounded with sequence, carry-over, or period effects. Therefore, appropriate tests were done to find significant confounding effects.

ANOVA

ANCOVA

When the data is normally distributed and no carry-over effects were detected, a dependent sample t-test was used. The usage of ANOVA will return the same decision for two groups as the t-test. If a non-parametric test had to be used, a Wilcoxon matched pairs test was used. In order to test whether the test results were robust with respect to disturbing factors, an analysis of covariance (ANCOVA) was conducted. The underlying assumptions for using ANCOVA (i.e., linear regression and homogeneity of the regression coefficients) were checked.

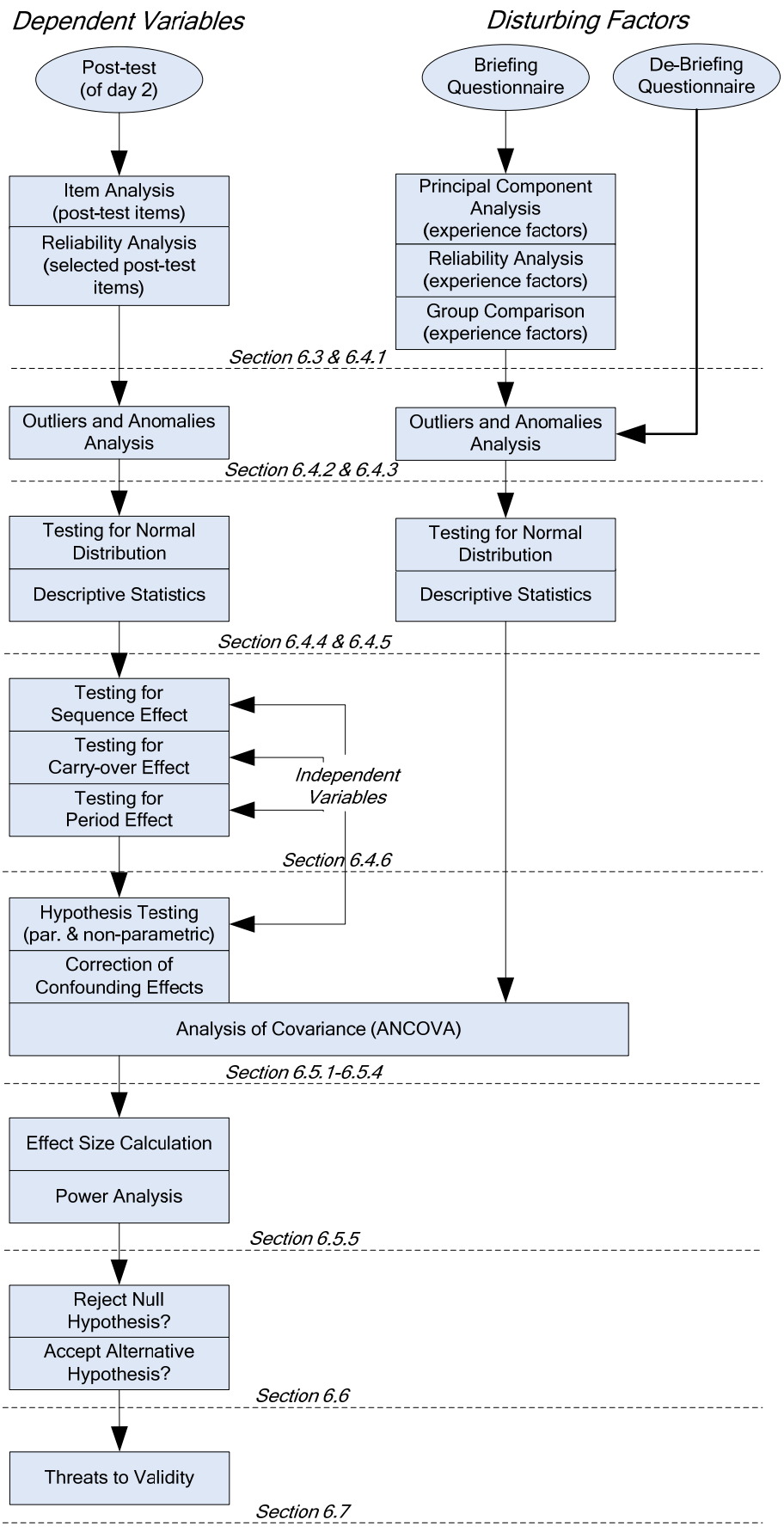


Figure 39 Data analysis procedure

A posteriori power analysis Effect size calculations

After conducting the hypothesis tests, the effect sizes were calculated and a power analysis was done. Statistical power analysis is used to test null hypotheses and to understand the strength of the results of an experiment (Cohen, 1988). Usually, research should perform a power analysis before running the experiment to ensure that the experimental design will find a statistically significant effect if it exists. The power of a statistical test is dependent on three different variables: significance level α, effect size d (based on t-test) or f (based on F-test), and number of subjects n. In this case an *a priori* power analysis was difficult because no effect size was known, since no similar studies are available from which an effect size could be taken. Therefore, an *a posteriori power analysis* for all hypotheses using the obtained effect size was done using the tool G*Power (ver. 3.0.5) (Faul, 2006). The tool calculates the power based on a given α, n, d (or f), and the applied hypothesis test.

Significance level $\alpha < 0.05$; power > 0.80

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. A final decision about the rejection of the null hypotheses, respectively the acceptance of the alternative hypotheses, was made.

7.3 Data Preparation

Before the results can be presented and hypothesis tests can be performed, this section describes how scales were derived from the data sets and how the reliability of some disturbing factors as well as the pre- and post test questionnaires was improved based on using principal component analysis (i.e., factor analysis) or item analysis. The reliability of the disturbing factor has an impact on the reduction of the measurement error in later analysis steps.

7.3.1 Principal Component Analysis of the Briefing Questionnaire

The principal component analysis is a variable reduction procedure. It serves to aggregate sets of items to a few factors. In this case, PCA has been applied to all items of the briefing questionnaire, which intend was to capture the experience level of the subjects. PCA has been used for identifying items of the briefing questionnaire, which measure the same construct of experience and which are not correlated (or very low correlation) to the other experience factors. The idea is a minimal set of items that account for the most of the variance of a factor. The PCA produces five scales, which describe five different dimensions, i.e., experience levels, as described in Section 7.1.2.3. The detailed procedure and results of the PCA can be found in Appendix C.1.

Cronbach alpha levels

The following table shows the *Cronbach's alpha* values, which are the reliability measure for the derived scales based on standardized values. Fisseni states that a Cronbach's alpha<0.80 should be considered as small, 0.80-0.90 as medium, and >0.90 as a high reliability (Fisseni, 1997). However, Cronbach's alpha depends on the number of items that belong to a scale: A higher number of items that positively correlate amongst each other produce higher reliability levels. The inter-item correlation values can be found in Appendix C.1.

Table 44 Reliability analysis of scales for experience levels

Disturbing Factor	Factor Name	Cronbach's Alpha Based on Standardized Items	N of Items
exp_dev	Experience Development	.720	5
exp_jp	Experience Java Programming	.845	4
exp_ref	Experience Refactoring	.888	4
exp_qa	Experience Software Quality Assurance	.896	4
exp_main	Experience Software Maintenance	.951	6

Medium or highly reliable scales have been produced

The results in the table show that the PCA produces almost medium or highly reliable scales for further statistical evaluation. Only the disturbing factor exp_dev has a lower Cronbach alpha, which is still acceptable.

7.3.2 Item Analysis of the Post-Test Questionnaire

Item selection based on: discrimination index discrimination coefficient item difficulty

An item analysis based on the post-test data from the second 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 I, 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 discrimination 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 following formula, where m is the number of possible answers to a specific question:

Item difficulty

$$p = \frac{\#right\_answers - (\frac{\#false\_answers}{m-1})}{\#answered\_total} * 100$$

The difficulty was corrected first by the fact that people could have answered correctly by chance and second, by the number of people who really answered the item (Bühner, 2006). The item difficulty has a range of [-100,100].

Reliability increased from 0.66 up to 0.81

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. More details about the intermediate results of the item analysis can be found in Appendix C.2. By doing this, the reliability of the test with 65 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.

Calculation of pre- & post-test scores

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. 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.

7.4 Experimental Results

This section presents the results of the experiment. First, a group comparison between the two groups was done to check whether there were significant differences between the two groups regarding their experience. After an outlier and anomalies analysis, first results are presented by using descriptive statistics. Afterwards, several sections are related to the investigation of confounding effects, such as period-, sequence, or carry-over effects, because these are mostly neglected in software engineering evaluations. The results of hypothesis tests are provided next. The last sections provide sample size calculations for future experiments, and describe the threats to validity.

7.4.1 Group Comparison on Experience Level

To further control differences between subjects, random assignment of the subjects to the two groups was done on the basis of four levels of

experience in order to reduce possible confounding effects. The following table shows the descriptives of the disturbing factors related to the different experience dimensions measured and whether they are normally distributed.

Table 45 Descriptive statistics of experience level variables

	N	Minimum	Maximum	Mean	Std. Deviation	Normal Distribution
Software Development Experience (exp_dev)	19	.00	1.00	.47	.31	Yes
Java Experience (exp_jp)	19	2.00	6.25	4.20	1.23	Yes
Refactoring experience (exp_ref)	19	1.00	4.75	2.08	1.22	No
SQA experience (exp_qa)	19	1.00	6.00	2.84	1.48	Yes
Software maintenance experience (exp_main)	19	1.00	5.83	2.61	1.55	No

Normality test

A test for normality was performed by using the Shapiro-Wilk test (Shapiro & Wilk, 1965), histograms, boxplots, and (normal and detrended) Q-Q plots. The detailed discussion and the outcome can be found in Appendix C.4.1. The factors exp_ref and exp_main are not normally distributed. Hence, parametric tests should not be applied and their results should be checked by performing non-parametric tests, if applied.

Independent sample t-test: significant group difference for exp_qa

An independent sample t-test was performed to check for significant differences between the two groups. The results in Table 46 show that no significant group difference could be found for almost all experience dimensions by comparing the means of the experimental and control groups. However, a significant difference (α=.047) for the disturbing factor exp_qa (experience software quality assurance) was found.

Table 46 Independent samples t-test for experience level equality

	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference
Software development experience (exp_dev)	.667	17	.514	.098	.147
Java Experience (exp_jp)	.282	17	.781	.164	.581
Refactoring experience(exp_ref)	-.197	17	.846	-.114	.577
SQA experience (exp_qa)	2.147	17	***.047***	1.336	.622
Software maintenance experience (exp_main)	.392	17	.700	.287	.733

exp_ref and exp_main are not normally distributed

As stated before, the factors exp_ref and exp_main are not normally distributed. Therefore, a Mann- Whitney U test for independent samples was conducted, which is equivalent to the Wilcoxon rank sum test and the Kruskal-Wallis test for two groups. The results for both variables

exp_ref and exp_main confirm the results of the parametric t-test.

Table 47 Non-parametric Mann-Withney U test for experience level equality

	Refactoring experience	Software maintenance experience
Mann-Whitney U	44.500	37.000
Wilcoxon W	89.500	82.000
Z	-.042	-.659
Sig. (2-tailed)	.967	.510

Significant group difference may be the cause for different carry-over effects

The group difference will play a role if the data from the second period cannot be used due to carry-over effects (see Section C.5.3). In that case, the intervariability between the subjects cannot be neglected, because repeated measure analysis is applied. In that case only the data from the first day can be used and the statistical testing has to be done as for a parallel design. Hence, special attention has to be paid to this significant difference between the control and experimental groups.

7.4.2 Outlier Analysis

Occasionally, one encounters data with one or more observations that deviate markedly from other observations in the sample. Such observations are called outliers. If an outlier is detected, it calls for detective work on the part of the researcher because there may be extreme values of the random variability or the result of deviations from prescribed experimental procedures, recording errors, etc. In the former instance, they would be processed in the same manner as the other observations. If some explanation can be found, one may replace the observation with new data, correct the observations if records permit, or reject the observation.

Outliers were detected mathematically and graphically

First indications of outliers were provided by higher values for the kurtosis of the variables. Outliers can be identified either mathematically or graphically. A simple mathematical approach is to convert the raw data into standardized values by a z-transformation. As a rule of thumb, values higher than +/-2.5 can be suspected of being outliers (i.e., these corresponds to values outside the 99% confidence interval). In this outlier analysis, data points with a z-value lower than +/-2.5 were also selected as outliers: By comparing these outliers with the data set mean and by looking at the box-and-whisker plot for the data sets, one can identify outliers more easily. Eight outliers have been detected (see Appendix C.3).

7.4.3 Anomaly Analysis

This section identifies and discusses anomalies in the data set such as missing values and undefined values for the dependent variables as well as for the disturbing factors.

Missing values

Regarding missing values, the number of missing values was reduced to four missing values by several measures: first reminding the subjects to check whether they had covered all the assignments and questions; second, by checking whether the time values had filled out by the subjects when they delivered their experiment material back to the evaluators, and third, by directly asking the subjects when values were missing (i.e., the material was checked immediately after delivery). Two of the missing values were due to a division by zero for the calculation of application accuracy (i.e., SPSS categorizes a division by zero as a missing value). In these cases, the values were replaced by the value zero. The two other missing values were found in the briefing questionnaire of one subject (question B3.1 and B3.2). The values were set to "0" (i.e., "0"=unexperienced, "1"=experienced) by an interpretation of the results of related questions because the subject was expected to have programmed a lower number of applications compared to subjects who had the value "1".

Undefined values

Undefined values were detected by using frequency tables created by SPSS. All the undefined values were related to errors from transferring the data from the paper-based questionnaires to the SPSS data sheets.

7.4.4 Descriptive Statistics for the Dependent Variables

The following tables show the descriptives of the dependent variables for each day and each group (i.e., experimental group/day 1; control group/day 1; experimental group/day 2; control group/day 2). Since extreme outliers were removed from the data set, the mean can be used to compare the performance of the two groups. For those variables that are not normally distributed, the median is used.

Table 48 Descriptive statistics of dependent variables (experimental group/day 1)

Dependent Variables	N	Min.	Max.	Mean	Median	Std. Dev.	Skewness	Kurtosis	Normal Distribution
ucorr	10	27.00	44.00	36.20	36.00	5.73	-.028	-.915	yes
know_diff	10	10	25	18.20	18.50	5.16	-.095	-1.015	yes
know_diff_remember	10	-1	11	6.90	8.50	3.98	-.870	-.090	yes
know_diff_understand	10	1	10	5.00	6.50	2.45	.454	1.226	yes
know_diff_apply	10	8	24	15.80	15.50	5.51	.303	-.933	yes
know_diff_analyze	10	-3	8	4.90	7.00	4.01	-1.044	-.124	***no***

Dependent Variables	N	Min.	Max.	Mean	Median	Std. Dev.	Skewness	Kurtosis	Normal Distribution
aeff	9	.115	.781	.446	.451	.221	-.126	.717	yes
acomp	10	.333	1.000	.658	.645	.230	.303	-.933	yes
aaccu	10	.300	1.000	.681	.670	.204	-.242	.167	yes

Table 49 Descriptive statistics of dependent variables (control group/day 1)

Dependent Variables	N	Min.	Max.	Mean	Median	Std. Dev.	Skewness	Kurtosis	Normal Distribution
ucorr	9	18.00	38.00	29.33	32.00	7.01	-.391	-1.347	yes
know_diff	9	1	11	7.44	8.00	3.61	-.751	-.461	yes
know_diff_remember	9	0	3	2.00	2.00	1.12	-.690	-.800	yes
know_diff_understand	9	-1	4	1.56	1.00	1.81	.210	-1.322	yes
know_diff_apply	9	2	12.25	6.78	6.50	3.29	.082	-.664	yes
know_diff_analyze	9	0	8	3.22	3.00	2.59	.688	-.148	yes
know_diff_create	9	0	2	.67	1.00	.70	.606	-.286	***no***
aeff	9	.091	.542	.274	.243	.150	.487	-.524	yes
acomp	9	.167	.708	.421	.417	.174	.033	-.688	yes
aaccu	9	.200	.775	.514	.500	.167	-.355	.614	yes

Table 50 Descriptive statistics of dependent variables (experimental group/day 2)

Dependent Variables	N	Min.	Max.	Mean	Median	Std. Dev.	Skewness	Kurtosis	Normal Distribution
ucorr	8	28.00	46.00	40.00	41.00	5.88	-1,214	1,848	yes
know_diff	9	8	26	17.67	20.00	7.40	-.092	-2.080	yes
know_diff_remember	8	1	8	5.13	5.50	2.17	-.774	-.923	yes
know_diff_understand	9	2	12	7.33	7.00	3.60	.043	-1.404	yes
know_diff_apply	9	5	17	9.917	8.00	4.64	.555	-1.194	yes
know_diff_analyze	9	0	8	4.44	6.00	2.55	-.467	-.688	yes
know_diff_create	9	0	2	1.22	1.00	.83	-.501	-1.275	***no***
aeff	9	.143	.703	.348	.333	.179	.891	.550	yes
acomp	9	.208	.708	.413	.333	.193	.555	-1.194	yes
aaccu	9	.446	1.000	.775	.807	.190	-.493	-.637	yes

Table 51 Descriptive statistics of dependent variables (control group/day 2)

Dependent Variables	N	Min.	Max.	Mean	Median	Std. Dev.	Skewness	Kurtosis	Normal Distribution
ucorr	10	24.00	42.00	33.80	33.00	6.60	-.035	-1.649	yes
know_diff	10	-6	10	4.50	6.00	4.83	-1.135	1.240	yes
know_diff_remember	8	1	3	1.63	1.50	.74	.824	-.152	yes
know_diff_understand	10	-4	7	1.70	2.00	3.02	-.186	.808	yes
know_diff_apply	10	2	16	7.700	7.62	4.26	.617	.200	yes
know_diff_analyze	10	-2	4	1.00	0.00	1.944	.45	-.516	yes
know_diff_create	9	-1	2	.44	0.00	.882	.21	.144	yes
aeff	10	.077	.394	.242	.252	.105	-.398	-.345	yes
acomp	9	.083	.510	.282	.270	.137	.082	-.664	yes
aaccu	10	.288	1.000	.626	.603	.220	.284	-.595	yes

Table 52 Descriptive statistics of dependent variables (inf_qua)

Dependent Variables	N	Min.	Max.	Mean	Median	Std. Dev.	Skewness	Kurtosis	Normal Distribution
inf_qua (LSEP)	19	3.80	6.60	5.2000	5.20	.16258	.117	-.189	yes
inf_qua (EP)	19	2.60	6.20	4.4211	4.40	.24177	.079	-.428	Yes

First day: Experimental group performs better than control group

As can be seen for the first day (i.e., tables marked with day 1), the subjects of the experimental group got higher mean values for all dependent variables that are normally distributed. The biggest differences were obtained for the knowledge acquisition difference variables, except for know_diff_create ($md_{control}$ = 1.00; $md_{experimental}$ = 1.00). Regarding efficiency (aeff), completeness (acomp), and accuracy (aaccu), the mean differences between the experimental and control groups were smaller, but the performance of the experimental group was still better than that of the control group.

Second day: Experimental group performs better than control group

For the second day (i.e., tables marked with day 2), the difference for understanding correctness (ucorr) were similar to the first day.

Regarding perceived information quality, it can be seen that the subject rated the quality higher for the learning spaces compared to the experience packages (M_{LSEP} = 5.20; M_{EP} = 4.42). The following table shows the relative differences between the two days.

Table 53 Relative improvement of the two two days (experimental vs. control group)

Dependent Variables	Day 1	Day 2	Average
ucorr	23.41 %	18.34 %	20.88 %
know_diff	144.62 %	292.67 %	218.65 %
know_diff_remember	245.00 %	214.72 %	229.86 %
know_diff_understand	220.51 %	331.18 %	275.84 %
know_diff_apply	133.04 %	28.79 %	80.91 %
know_diff_analyze	52.17 %	344.00 %	198.09 %
know_diff_create	64.18 %	177.27 %	120.73 %
aeff	62.61 %	43.38 %	53.00 %
acomp	56.26 %	46.52 %	51.39 %
aaccu	32.57 %	23.83 %	28.20 %
inf_qa	-	-	17.62 %

The results illustrate that all improvements are positive, which means that the experimental group performed better than the control group. On the one hand, for understanding correctness as well as for all the application variables (i.e., efficiency, completeness, and accuracy), quite similar improvements were achieved for both days. On the other hand, very different improvement rates were obtained for the knowledge acquisition difference variables.

Discussion of improvement rates

Even if the subjects did not rate the complexity of the second day's experience packages higher than that of the first day, they performed slightly worse regarding the application variables. A much higher improvement was achieved for the overall knowledge acquisition difference for the second day, which was due mainly to the higher score for the levels understand, analyze, and create. Higher scores for the level understand mean that the subjects were able to build relationships between basic refactoring concepts and to construct cognitive if-then rules (i.e., conceptual knowledge) especially during the second day. In order to achieve higher scores for the levels analyze and create, the subjects needed more contextual knowledge in refactoring itself, which means that they needed to apply lower levels of knowledge (i.e., factual and conceptual knowledge) in practice, respectively during the experiment's assignments. Hence, effects confounding the treatment effect (e.g., period effect, carry-over effect, etc.) should be expected (see Section 7.4.6 for their investigation). The perceived information quality rose by 17.6%.

Statistical tests are necessary to check whether the differences are statistically significant

Statistical tests of the dependent variables have to confirm whether the mean differences are statistically significant or not. A test for normality was performed by using the Shapiro-Wilk test (Shapiro & Wilk, 1965) together with the graphical analysis of the histograms, boxplots, and (normal and detrended) Q-Q plots. The detailed results can be found in Appendix C.4.1.3. Kurtosis is a measure of whether the data are peaked or flat relative to a normal distribution. Skewness is a measure of symmetry, i.e., the lack of symmetry. Especially the variable know_diff_create is not normally distributed for three data sets out of four. Furthermore, the variable know_diff_analyze (experimental group, day 1) is not normally distributed. It can be observed that the data sets of these variables have high standard deviations compared to their means. In addition, the high values for skewness and kurtosis give hints about high peaks, respectively asymmetry, in the distribution. Hence, parametric tests should not be applied and their results should be checked by performing non-parametric tests, if applied.

7.4.5 Descriptive Statistics for the Disturbing Factors

The following table shows the descriptives of the disturbing factors which were measured by means of the briefing and debriefing questionnaires.

Table 54 Descriptive statistics of disturbing factors

Disturbing Factors	N	Min.	Max.	Mean	Std. Dev.	Skewness	Kurtosis	Normal Distribution
exp_dev	19	.00	1.00	.47	.31	.079	-.780	yes
exp_jp	19	2.00	6.25	4.20	1.23	-.102	-1.150	yes
exp_ref	19	1.00	4.75	2.08	1.22	1.113	.022	***no***
exp_qa	19	1.00	6.00	2.84	1.48	.643	-.394	yes
exp_main	19	1.00	5.83	2.61	1.56	.750	-.594	***no***
tn	19	.00	.60	.19	.23	.631	-1.200	***no***
pre-test	19	9.00	32.00	19.84	7.19	.258	-1.025	yes
inf_qua (LSEP)	19	3.80	6.60	5.20	.16	.117	-.189	yes
inf_qua (EP)	19	2.60	6.20	4.42	.24	.079	-.428	Yes

Subjects' experience

The details of the normality test can be found in Appendix C.4.1. The subjects possessed an average experience level in software development (M = .47, SD = .31; 0=no experience, 1=expert) and java programming (M = 4.20, SD = 1.23), low experience in refactoring (M = 2.08, SD = 1.22), quality assurance (M = 2.84, SD = 1.48), and software maintenance (M = 2.61, SD = 1.56) where "0" corresponds to no experience and "6" refers to expert level. As can be seen from the table, most of

the subjects had enough time to work through the information, to solve the assignments, and become familiar with the Wiki and the learning spaces (0=enough time; 1=lack of time). An interesting observation is that the subjects rated the information quality of the experience packages (M = 4.42, SD = 1.06) lower than that of the learning spaces (M = 5.20, SD = .71). The pre-test factor was normally distributed as two thirds of the other factors. However, exp_ref, exp_main, and tn were significantly different from a normal distribution. Using the disturbing factors as a covariate in ANOVA will show whether the covariate correlates with the treatment effect and hence can reduce the measurement error.

7.4.6 Confounding Effects Impacting the Treatment Effect

In software engineering the investigation of confounding effects is mostly neglected

The design of this experiment is a within-subject counterbalanced design. In clinical research, this design is mostly called two-period crossover design or simply AB|BA design. Besides the reduced sample size to achieve a specified power, more statistical power because of paired comparison, and the elimination of between-subject variability compared to a one-period parallel design, several risks of effects confounding the treatment effect need to be considered.

Randomization and counterbalacing are not measures for preventing confounding effects

Most research work in software engineering that applies a crossover design for experiments, does not consider any investigation of the confounding effects. Randomization and counterbalancing have been stated as ways to prevent such effects. However, the literature survey on effect terminology and statistical methods for investigating and/or correcting them (see Appendix C.5) confirmed that these effects must be considered. A lot of research has been done on investigating these effects in areas other than software engineering. Many approaches have been developed and all of them have been criticized to some extent. It would be outside the scope of this work to analyze and compare the available statistical procedures in detail for their usage in software engineering. However, it seems that software engineering research still applies statistical approaches (e.g., testing for carry-over effects) that have been considered wrong by renowned statisticians for more than twenty years. The detailed and very complex analysis of the confounding effects can be found in Appendix C.5 and its subsequent chapters. The following table provides the summary of the confounding effects that were detected by using appropriate techniques (see Appendix C.5).

Table 55 Overview of confounding effects

	Period Effect detected	Sequence Effect detected	Position Effect detected	Carry-Over Effect detected
ucorr	no (yes with low power)	no	no (yes with low power)	no
know_diff	no	no	no	no
know_diff_remember	no	no	no	no
know_diff_understand	no	no	no	no
know_diff_apply	**yes**	no	**yes**	no
know_diff_analyze	no	no	no	no
know_diff_create	no	no	no	no
aeff	no	no	no	no
acomp	**yes**	no	**yes**	no
aaccu	no	no	no	no

Period effects (i.e., hypothesis testing with ind. T-test/Mann Whitney U test and mean cross-offer difference) and position effects (i.e., hypothesis testing with repeated measures ANOVA) were found for the same variables. By comparing Table 103, Table 105, and Table 107 with Table 110 in Appendix C.5, it can be seen that the results are exactly the same. This means that period effects are the same as position effects (i.e., interaction of treatment*sequence effects) and that we can use the procedure of Hills and Armitage (Hills & Armitage, 1979) to correct for period effect respectively for position effects, in the subsequent sections.

7.5 Hypothesis Testing

The analysis methods used in the subsequent sections are based on dependent samples because of the repeated within-subjects design. For those cases where period effects were detected, specific methods were used for correction. In the previous sections of the descriptive statistics, we investigated whether the data from two groups and the two days are distributed normally. These tests for normality are relevant for those tests where only the data from the first day can be used due to the confounding effects. Senn states that in analyzing the data from crossover trials, we have to expect that the crossover differences are distributed at random across the true treatment effect. For the t-statistic to be an efficient way of examining uncertainty about the treatment effect we should also believe that the data are distributed normally, although this assumption is less important since the t-test is quite robust against normality violations. Nevertheless, the test for normality has also been performed for dependent variables based on the differences between the two periods. For the variables ucorr (strong deviation from a normal distribution), know_diff_understand (medium deviation), and know_diff_create (difficult to confirm the deviation due to a small number of different values),

parametric tests should only be performed in combination with non-parametric tests.

In each of the subsequent sections, simple hypothesis tests are used as well as methods that consider covariates (i.e., disturbing factors) for reducing the measurement error.

Assumptions to be met before AN(C)OVA can be used

Several assumptions have to be fulfilled when an analysis of covariance (ANCOVA) is executed, which are unfortunately not checked in most empirical research: An analysis of variance (ANOVA) requires that the measurements are independent and that random sampling has been done. In addition, the homogeneity of variance and normality must be fulfilled. The complex statistical procedures and the results are elaborated in Appendix C.6.

A violation of the assumptions leads to progressive decisions in favor of the alternative hypothesis

When the previously named assumptions are not fulfilled, progressive decisions are obtained in favor of accepting the alternative hypothesis and other risk exist (Buser, 1995). For example, the more the assumption of "equality of regressions" is violated, the more conservative ANCOVA becomes (increased likelihood of Type II errors: thinking there is no relationship when, in fact, there is a relationship).

For the further application of ANCOVA, it is important to know that first, disturbing factors with low reliability reduce the power of ANCOVA (Bortz & Döring, 2001) and can lead to contortions of the corrected treatment effect .

Covariates with low reliability reduce power of ANCOVA

As a rule of thumb, covariates should have a reliability coefficient of about .80 or higher. Otherwise, "one will end up potentially adjusting sampling error with measurement error, and creating a mess" (Loftin & Madison, 1991). Second, the more covariates are integrated into the ANCOVA model, the greater the likelihood that an additional covariate will have little residual correlation with the dependent variable after other covariates are controlled. The marginal gain in explanatory power is offset by the loss of statistical power (a degree of freedom is lost for each added covariate). Third, the disturbing factors have an interval scale, and are assumed to be measured without error. Imperfect measurement reduces the statistical power of significance tests for ANCOVA, and for experimental data, there is a conservative bias (increased likelihood of Type II errors: thinking there is no relationship when, in fact, there is a relationship). The following table summarizes which disturbing variables will be used for ANCOVA, including their reliability:

Table 56 Disturbing variables suitable for ANCOVA

Disturbing variable / Dep. variable	exp_dev	exp_jp	exp_ref	exp_sqa	exp_main	tn	pre-test	inf_qua_LSEP	inf_qua_EP
ucorr		yes	yes	yes	yes		yes		
ut									
know_diff					yes			(yes)	
know_diff_remember								(yes)	
know_diff_understand									
know_diff_apply		yes							
know_diff_analyze						(yes)			yes
know_diff_create									
aeff				yes					
acomp		yes							
aaccu	yes								
reliability	.720	.845	.888	.896	.951	.602	.787	.660	.753

Cells with a yes in parentheses mark a disturbing factor with low reliability. It has been decided not to apply ANCOVA for these variables.

First interpretations were made on the basis of the descriptive statistics (Section 7.4.3 and Section 7.4.5). The next sections will describe the results of statistical hypothesis tests.

7.5.1 Hypothesis $H_{1.1}$ (Understandability)

7.5.1.1 Hypothesis $H_{1.1}$ (Understanding Correctness)

$H_{1.1}$ $\mu(ucorr_{LSEP}) > \mu(ucorr_{EP})$

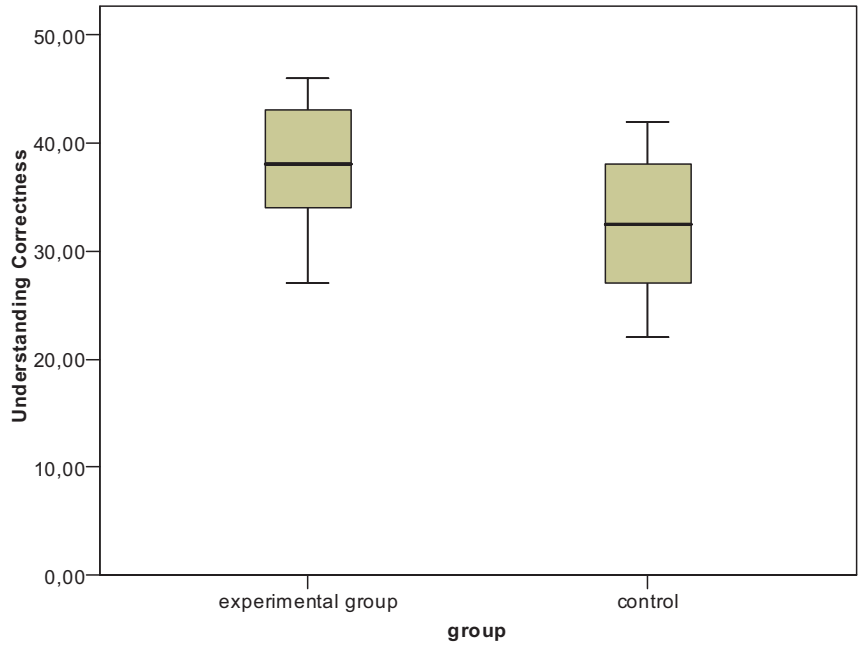


Figure 40 Box-and-whisker plot for understanding correctness (ucorr)

Table 57 One-tailed dependent sample t-test for understanding correctness (ucorr)

	Pair Differences					df	t	Crit. $T_{0..95}$	p-value
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference lower	upper				
ucorr	5.444	6.419	1.513	2.252	8.636	17	3.599	1.740	.002

The paired-samples t-test revealed significant differences in understanding correctness between the experimental and the control group, t(17) = 3.60, p = .002. The mean differences for ucorr were not normally distributed. A Wilcoxon matched paired test provided the same result as the t-test (p = .002).

The ANOVA for repeated measures without using covariates provides the same result as the one-tailed dependent sample t-test (F(1,17) = 12.95, *p* = .002), which is normal because only two groups were compared. According to Table 56 the disturbing factors exp_jp, exp_ref, exp_sqa, exp_main, and pre-test fulfill the assumptions for using them to correct the measurement error of ucorr.

Table 58 ANCOVA results for understanding correctness (ucorr)

Dependent Variable	Covariate	F	Hypothesis df	Error df	p-value	Partial Eta Squared	Observed Power
ucorr	none	12.95	1	17	.002	.432	.923
	exp_jp	6.23	1	16	.024	.280	.649
	exp_ref	8.301	1	16	.011	.342	.772
	exp_sqa	10.36	1	16	.005	.393	.856
	exp_main	8.64	1	16	.010	.351	.788
	pre-test	2.677	1	16	.121	.143	.337

No significant interaction effect between ucorr and the disturbing factor was found for exp_jp (p = .127), exp_ref (p = .239), exp_sqa (p = .104), exp_main (p = .217), and pre-test (p = .598). Furthermore, it can be seen that for none of the covariates, the partial eta square (i.e., the proportion of the effect + error variance that is attributable to the effect: $\eta_p^2 = SS_{effect} / (SS_{effect} + SS_{error})$) increased. Partial eta square is a measure of association of the sample. Hence, the 18 subjects in the experimental group (*M* = 37.89, *SD* = 5.95) had a significantly better *understanding correctness* compared to the 18 subjects of the control group (*M* = 32.44, *SD* = 6.33), with F(1,17)=12.95, p = .002.

7.5.2 Hypothesis $H_{1.2}$ (Knowledge Acquisition)

7.5.2.1 Hypothesis $H_{1.2.1}$ (Knowledge Acquisition Difference)

$H_{1.2.1}$ $\mu(know\_diff_{LSEP}) > \mu(know\_diff_{EP})$

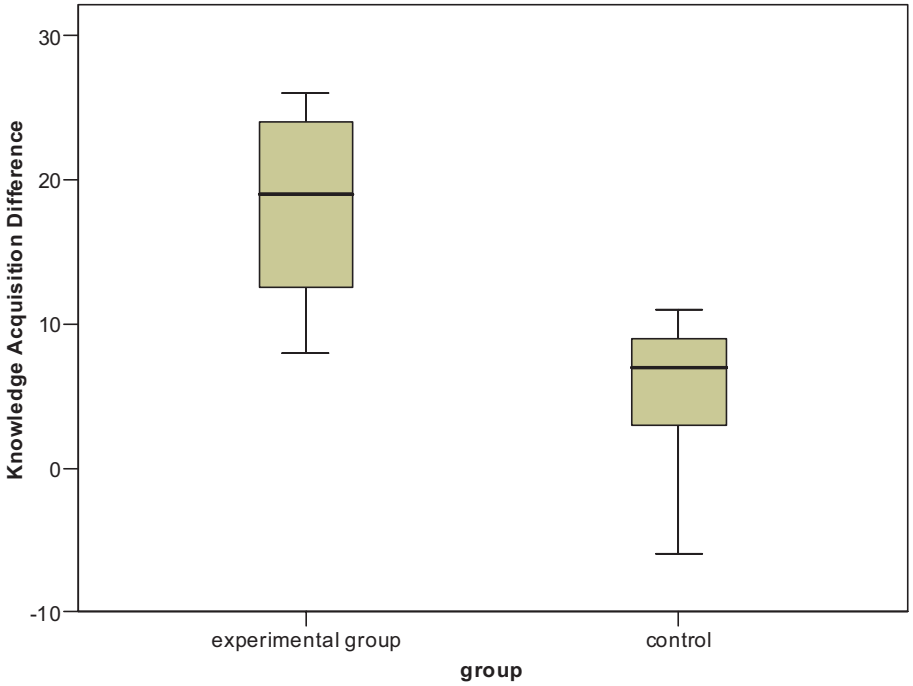


Figure 41 Box-and-whisker plot for knowledge acquisition difference (know_diff)

Table 59 One-tailed dependent sample t-test for knowledge acquisition difference (know_diff)

	Pair Differences					df	t	Crit. $T_{0..95}$	p-value
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference lower	upper				
know_diff	12,053	7,329	1,681	8,520	15,585	18	7.168	1.734	.000

The paired-samples t-test revealed significant differences in knowledge acquisition difference between the experimental and control group, t(18) = 7.17, p = .000.

According to Table 56 the disturbing factors exp_main, and inf_qua_LSEP (remark: low reliability) fulfill the assumptions for using them to correct the measurement error of know_diff. The following table shows the results of ANCOVA only for the disturbing factor exp_main.

Table 60 ANCOVA results for knowledge acquisition difference (know_diff)

Dependent Variable	Covariate	F	Hypothesis df	Error df	p-value	Partial Eta Squared	Observed Power
know_diff	none	51.38	1	18	.000	.741	1.000
	exp_main	6.65	1	17	.020	.281	.681

No significant interaction effect between knowledge acquisition difference and the disturbing factor was found for exp_main (p = .233). Furthermore, it can be seen that due to the covariate, the partial eta square increased after introducing it into the model.

Hence, the 19 subjects in the experimental group (*M* = 17.95, *SD* = 6.14) had a significantly better *knowledge acquisition difference* compared to the 19 subjects of the control group (*M* = 5.89, *SD* = 4.45), with F(1,18) = 51.38, p = .000.

7.5.2.2 Hypothesis $H_{1.2.2}$ (Knowledge Acquisition Difference on Cog. Levels)

Hypothesis $H_{1.2.1}$ has been refined further with regard to the different cognitive processes:

$$H_{1.2.2.X} \quad \mu(know\_diff\_x_{LSEP}) > \mu(know\_diff\_x_{EP})$$

In the following, the results of the hypothesis test for the cognitive dimension remember will be presented.

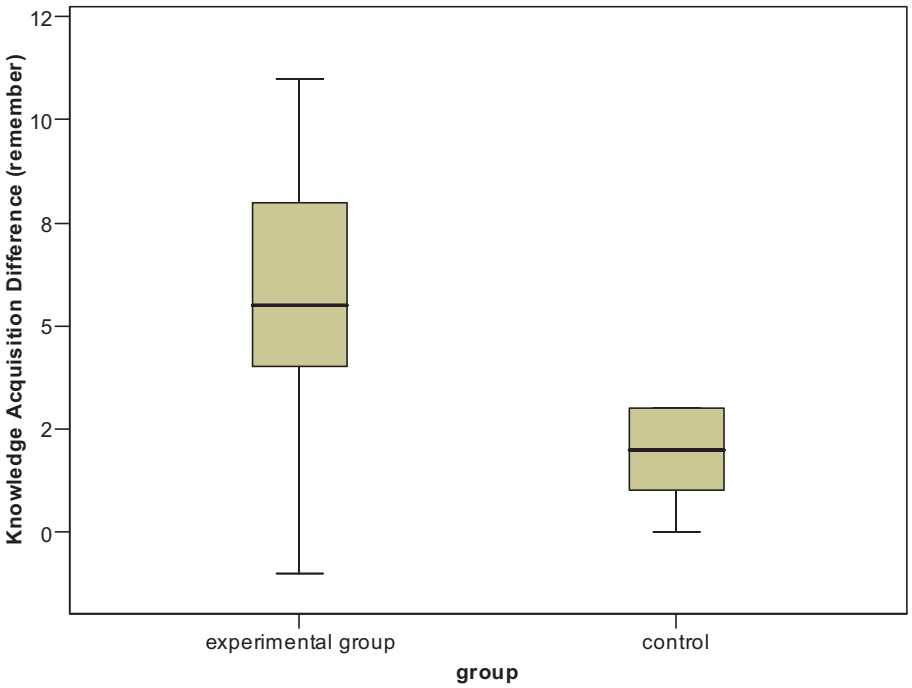


Figure 42 Box-and-whisker plot for knowledge acquisition difference remember (know_diff_remember)

Table 61 One-tailed dependent sample t-test for knowledge acquisition difference remember (know_diff_remember)

	Pair Differences					df	t	Crit. $T_{0..95}$	p-value
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference lower	upper				
know_diff_remember	3.813	3.710	.927	1.836	5.789	15	4.111	1.753	.001

The paired-samples t-test revealed a significant difference in the knowledge acquisition difference remember between the experimental and the control group, t(15) = 4.11, p = .001.

According to Table 56, the disturbing factor inf_qua_LSEP fulfills the assumptions for using it to correct the measurement error of know_diff_remember. However, due to the low reliability of .660, ANCOVA was not applied.

Hence, the 16 subjects in the experimental group (*M* = 5.63, *SD* = 3.20) had a significantly better *knowledge acquisition difference remember* compared to the 16 subjects of the control group (*M* = 1.81, *SD* = .98), with t(15) = 4.11, p = .001.

In the following, the results of the hypothesis test for the cognitive dimension understand will be presented.

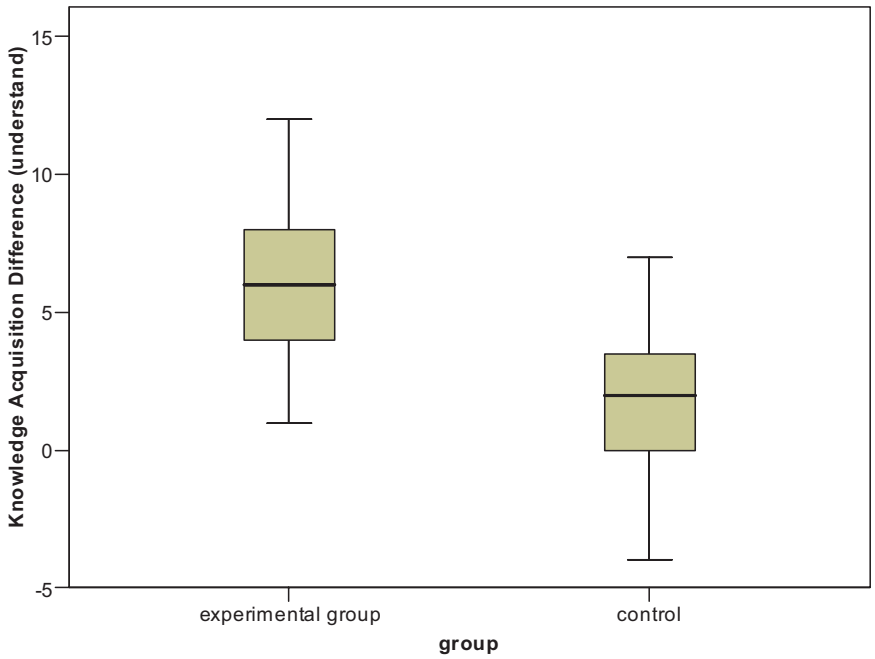


Figure 43 Box-and-whisker plot for knowledge acquisition difference understand (know_diff_understand)

Table 62 One-tailed dependent sample t-test for knowledge acquisition difference understand (know_diff_understand)

	Pair Differences					df	t	Crit. $T_{0..95}$	p-value
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference lower	upper				
know_diff_understand	4.474	4.195	.962	2.452	6.496	18	4.649	1.734	.000

The paired-samples t-test revealed a significant difference in the knowledge acquisition difference understand between the experimental and the control group, t(18) = 4.65, p = .000. The mean differences for know_diff_understand were not normally distributed. A Wilcoxon matched paired test provided p = .001, which is still very significant. Hence, a significant difference between the means of LSEP and EP has been found – the null hypothesis can be rejected.

According to Table 56 the disturbing factor inf_qua_LSEP fulfills the assumptions for using it to correct the measurement error of know_diff_remember. However, due to the low reliability of .660, ANCOVA was not applied.

Hence, the 19 subjects in the experimental group (*M* = 6.11, *SD* = 3.20) had a significantly better *knowledge acquisition difference understand* compared to the 19 subjects of the control group (*M* = 1.63, *SD* = 2.45), t(18) = 4.65, p = .000. In the following, the results of the hypothesis test for the cognitive dimension apply will be presented.

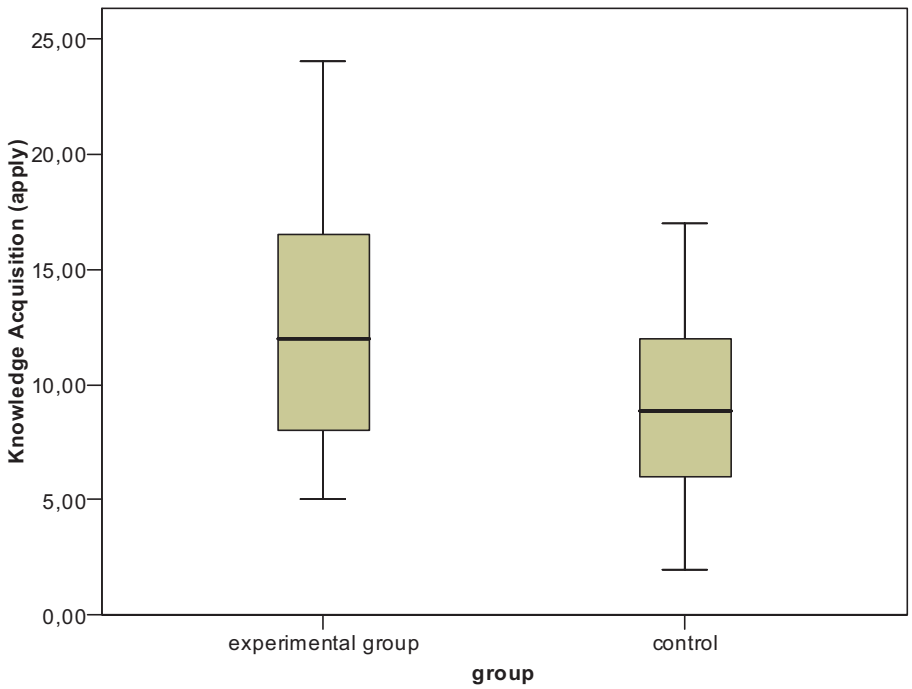


Figure 44 Box-and-whisker plot for knowledge acquisition difference apply (know_diff_apply)

Table 63 One-tailed dependent sample t-test for knowledge acquisition difference apply (know_diff_apply)

	Pair Differences					df	t	Crit. $T_{0..95}$	p-value
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference lower	upper				
know_diff_apply	3.958	6.931	1.634	.512	7.405	17	2.423	1.740	.027

The paired-samples t-test revealed a significant difference in the knowledge acquisition difference apply between the experimental and the control group, t(17) = 2.42, p = .027.

A period effect was detected for this variable in Section C.5.4. Table 64 shows the result.

Table 64 Two-tailed independent sample t-test for knowledge acquisition difference apply with period effect correction (know_diff_apply)

	Levene's Test for Equality of Variances		t-test for Equality of Means						
	F	p-value	t	df	p-value	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference Upper	Lower
know_diff_apply	1.597	.224	3.132	16	.006	8.306	2.652	2.68	13.93

It can be seen that the p-value decreases to p = .006, which is more significant than the t-test without correction for period effects.

According to Table 56, the disturbing factor exp_jp fulfills the assumptions for using it to correct the measurement error of know_diff_apply. The following table shows the results of ANCOVA.

Table 65 ANCOVA results for knowledge acquisition difference apply with period effect correction (know_diff_apply)

Dependent Variable	Covariate	F	Hypothesis df	Error df	p-value	Partial Eta Squared	Observed Power
know_diff_apply	none	5.87	1	17	.027	.257	.627
	exp_jp	.341	1	16	.567	.021	.085

No significant interaction effect between know_diff_apply and the disturbing factor exp_jp was found for exp_jp (p = .201). Furthermore, it can be seen that for the covariate, the partial eta square was reduced almost to zero, which means that the influence of this covariate can be ignored.

Hence, the 18 subjects in the experimental group (*M* = 12.40, *SD* = 5.33) had a significantly better *knowledge acquisition difference apply* compared to the 18 subjects of the control group (*M* = 8.44, *SD* = 4.03), with t(16) = 3.13, p = .006.

In the following, the results of the hypothesis test for the cognitive dimension analyze will be presented.

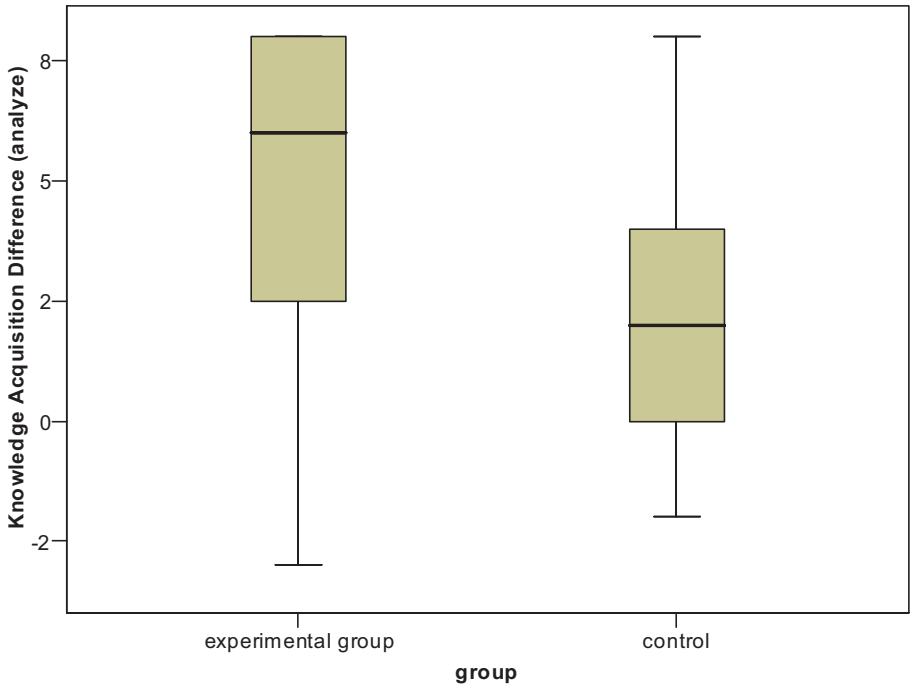


Figure 45 Box-and-whisker plot for knowledge acquisition difference analyze (know_diff_analyze)

Table 66 One-tailed dependent sample t-test for knowledge acquisition difference analyze (know_diff_analyze)

	Pair Differences					df	t	Crit. $T_{0..95}$	p-value
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference lower	upper				
know_diff_analyze	2.632	3.451	.792	.968	4.295	18	3.324	1.734	.004

The paired-samples t-test revealed a significant difference in the knowledge acquisition difference analyze between the experimental and the control group, t(18) = 3.32, p = .004.

According to Table 56, the disturbing factor tn fulfills the assumptions for using it to correct the measurement error of know_diff_analyze. However, tn had a low reliability of .602 and was not used to correct the measurement error of know_diff_analyze.

Hence, the 19 subjects in the experimental group (*M* = 4.68, *SD* = 3.32) had a significantly better *knowledge acquisition difference analyze* com-

pared to the 19 subjects of the control group (M = 2.05, SD = 2.49), with t(18) = 3.32, p = .004.

In the following, the results of the hypothesis test for the cognitive dimension create will be presented.

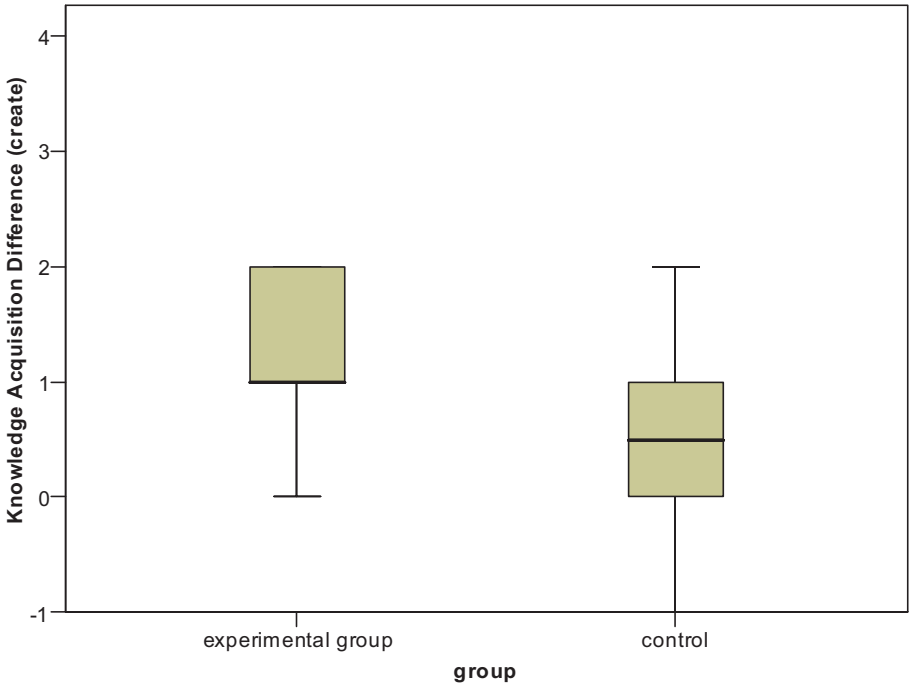


Figure 46 Box-and-whisker plot for knowledge acquisition difference create (know_diff_create)

Table 67 One-tailed dependent sample t-test for knowledge acquisition difference create (know_diff_create)

	Pair Differences					df	t	Crit. $T_{0..95}$	p-value
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference lower	upper				
know_diff_create	.667	1.188	.280	.076	1.258	17	2.380	1.740	.029

The paired-samples t-test revealed a significant difference in the knowledge acquisition difference create between the experimental and the control group, t(17) = 2.38, p = .029.

The mean differences for know_diff_create were not normally distributed. A Wilcoxon matched paired test provided p = .026, which is still very significant. Hence, a significant difference between the means of LSEP and EP has been found – the null hypothesis can be rejected.

According to Table 56 none of the disturbing factors fulfills the assumptions for using them to correct the measurement error of know_diff_create. Therefore, no ANCOVA was performed.

Hence, the 18 subjects in the experimental group (M = 1.22, SD = 1.00) had a significantly better *knowledge acquisition difference create* compared to the 18 subjects of the control group (M = 0.56, SD = .78), with t(18) = 2.38, p = .029.

7.5.3 Hypothesis $H_{1.3}$ (Application)

The effect of a learning space on an application was investigated by measuring application efficiency, application completeness, and application accuracy.

7.5.3.1 Hypothesis $H_{1.3.1}$ (Application Efficiency)

$H_{1.3.1}$ $\mu(aeff_{LSEP}) > \mu(aeff_{EP})$

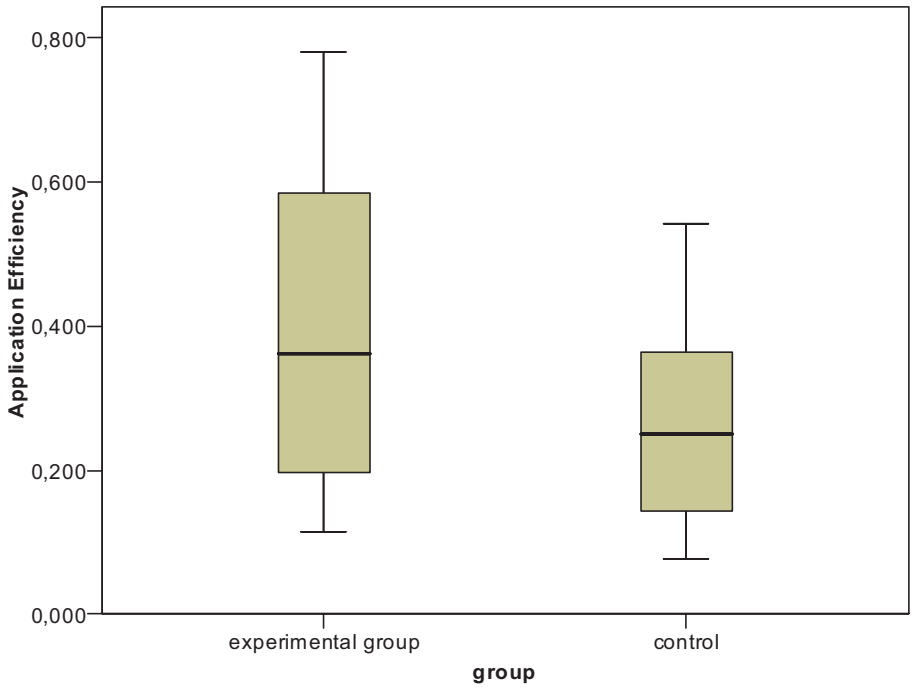


Figure 47 Box-and-whisker plot for application efficiency (aeff)

Table 68 One-tailed dependent sample t-test for application efficiency (aeff)

	Pair Differences					df	t	Crit. $T_{0..95}$	p-value
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference lower	upper				
aeff	.139	.186	.0438	.0465	.232	17	3.171	1.740	.006

The paired-samples t-test revealed a significant difference in application efficiency between the experimental and the control group, t(17) = 3.17, p = .006. The mean differences for aeff were not normally distributed. A Wilcoxon matched paired test provided p = .008, which is still very significant.

According to Table 56, the disturbing factor exp_sqa fulfills the assumptions for using it to correct the measurement error of aeff.

Table 69 ANCOVA results for application efficiency (aeff)

Dependent Variable	Covariate	F	Hypothesis df	Error df	p-value	Partial Eta Squared	Observed Power
aeff	none	10.06	1	17	.006	.372	.848
	exp_sqa	.031	1	16	.861	.002	.053

No significant interaction effect between aeff and the disturbing factor was found for exp_sqa (p = .153). Furthermore, it can be seen that for this covariate, the partial eta square decreased. Therefore, this covariate should be ignored.

Hence, the 18 subjects in the experimental group (*M* = .39, *SD* = .20) had a significantly better *application efficiency* compared to the 18 subjects of the control group (*M* = .26, *SD* = .13), with F(1,17) = 10.06, p = .006.

The calculation of efficiency was based on the *application time*, i.e., the time the subjects needed to solve the assignments. Since the total time (i.e., reading time plus application time) was fixed to 60 minutes, the subjects of the experimental group had less time to solve the assignments because they needed more reading time to get through the information provided by a learning space. This fact makes it harder for the experimental group to achieve higher values for application efficiency.

7.5.3.2 Hypothesis $H_{1.3.2}$ (Application Completeness)

$H_{1.3.2}$ $\mu(acomp_{LSEP}) > \mu(acomp_{EP})$

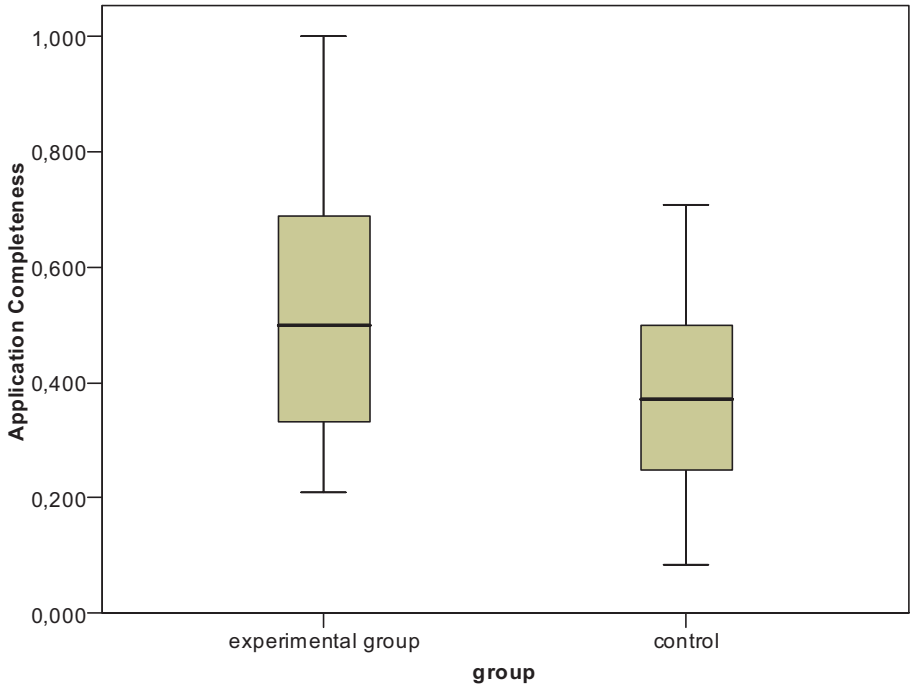


Figure 48 Box-and-whisker plot for application completeness (acomp)

Table 70 One-tailed dependent sample t-test for application completeness (acomp)

	Pair Differences					df	t	Crit. $T_{0..95}$	p-value
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference lower	upper				
acomp	.165	.289	.068	.021	.309	17	2.423	1.740	.027

The paired-samples t-test revealed a significant difference in application completeness between the experimental and the control group, t(17) = 2.42, p = .027. A t-test was used to adjust for the period effects:

Table 71 Two-tailed independent sample t-test for application completeness with period effect correction (acomp)

	Levene's Test for Equality of Variances		t-test for Equality of Means						
								95% Confidence Interval of the Difference	
	F	p-value	t	df	p-value	Mean Difference	Std. Error Difference	Upper	Lower
acomp	1.597	.224	3.132	16	.006	.346	.110	.11	.58

It can be seen that the p-value decreases to 0.06, which is more significant than the t-test without correction for period effects.

According to Table 56 the disturbing factor exp_jp fulfills the assumptions for using it to correct the measurement error of know_diff_apply. The following table shows the results of ANCOVA.

Table 72 ANCOVA results for application completeness with period effect correction (acomp)

Dependent Variable	Covariate	F	Hypothesis df	Error df	p-value	Partial Eta Squared	Observed Power
acomp	none	5.87	1	17	.027	.257	.627
	exp_jp	.341	1	16	.567	.021	.085

No significant interaction effect between acomp and the disturbing factor exp_jp was found for exp_jp (p = .201). Furthermore, it can be seen that for the covariate, the partial eta square was reduced almost to zero, which means that the influence of this covariate can be ignored.

Hence, the 18 subjects in the experimental group (*M* = .52, *SD* = .22) had a significantly better *application completeness* compared to the 18 subjects of the control group (*M* = .35, *SD* = .17), with t(16) = 3.13, p = .006 (i.e., the result of the corrected period effect was used).

7.5.3.3 Hypothesis $H_{1.3.3}$ (Application Accuracy)

$H_{1.3.3}$ $\mu(aaccu_{LSEP}) > \mu(aaccu_{EP})$

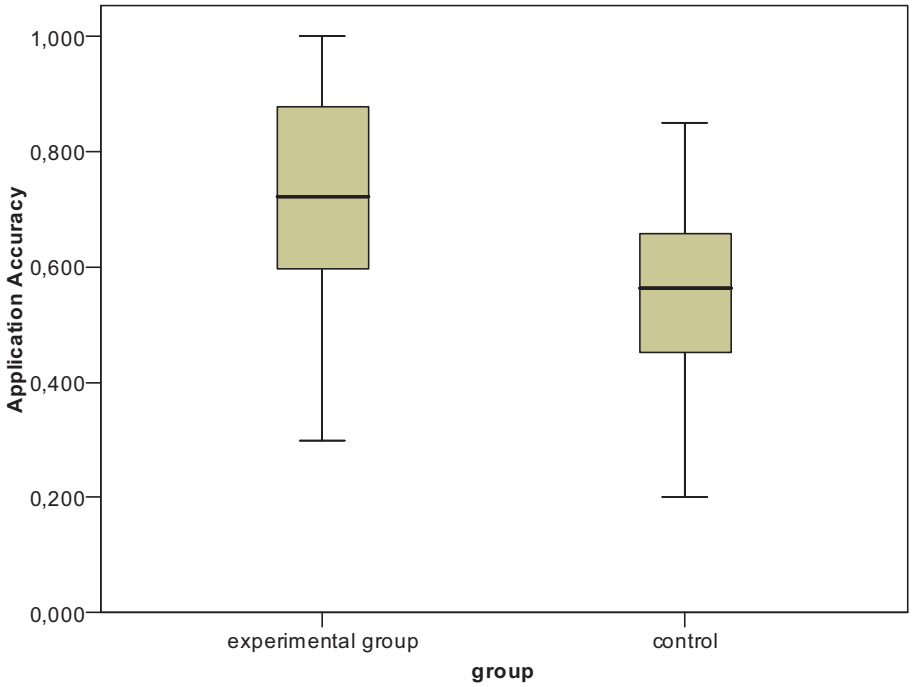


Figure 49 Box-and-whisker plot for application accuracy (aaccu)

Table 73 One-tailed dependent sample t-test for application accuracy (aaccu)

	Pair Differences					df	t	Crit. $T_{0..95}$	p-value
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference					
				lower	upper				
aaccu	.153	.316	.0724	.000732	.304907	18	2.111	1.734	.049

The paired-samples t-test revealed a significant difference in application accuracy between the experimental and the control group, t(18) = 2.11, p = .049.

According to Table 56 the disturbing factor exp_dev fulfills the assumptions for use it to correct the measurement error of know_diff_apply. The following table shows the results of ANCOVA.

Table 74 ANCOVA results for application accuracy (aaccu)

Dependent Variable	Covariate	F	Hypothesis df	Error df	p-value	Partial Eta Squared	Observed Power
aaccu	none	4.46	1	18	.049	.198	.515
	exp_dev	.072	1	17	.792	.004	.057

No significant interaction effect between aaccu and the disturbing factor exp_jp was found for exp_dev (p = .392). Furthermore, it can be seen that for the covariate, the partial eta square was reduced almost to zero, which means that the influence of this covariate can be ignored.

Hence, the 19 subjects in the experimental group (*M* = .72, *SD* = .18) had a significantly better *application accuracy* compared to the 19 subjects of the control group (*M* = .57, *SD* = .20), with t(18) = 2.11, p = .049.

7.5.4 Hypothesis $H_{1.4}$ (Perceived Information Quality)

$H_{1.4}$ $\mu(\text{inf}\_qua_{LSEP}) > \mu(\text{inf}\_qua_{EP})$

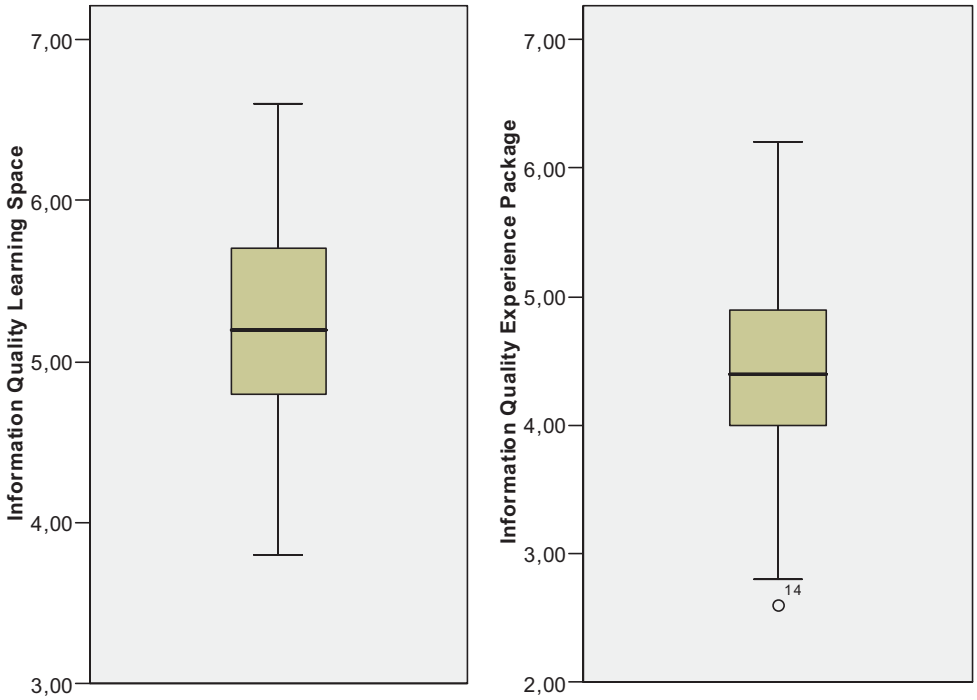


Figure 50 Box-and-whisker plot for information quality (LSEP and EP)

Table 75 One-tailed dependent sample t-test for perceived information quality (inf_qua)

	Pair Differences					df	t	Crit. $T_{0..95}$	p-value
	Mean	Std. Deviation	Std. Error Mean	95% Confidence Interval of the Difference lower	upper				
inf_qua	.77895	1.22546	.28114	.18829	1.36960	18	2.771	1.734	.013

The paired-samples t-test revealed a significant difference in the perceived information quality between the learning space and the experience package, t(18) = 2.77, p = .013.

Hence, the 18 subjects perceived the *perceived information quality* (*M* = 5.20, *SD* = .71) of a learning space significantly higher than the perceived information quality of an experience package (*M* = 4.42, *SD* = 1.05).

7.5.5 Effect Size Calculations and Power Analysis

Effect size is the strength of the relationship between two variables

The term "effect size" refers to the magnitude of the effect under the alternative hypothesis and describes the strength of a relationship between two variables. Effect sizes can be interpreted in terms of the percent of non-overlap of the experimental group's scores with those of the control group. An effect size of 0.0 indicates that the distribution of

scores for the experimental group overlaps completely with the distribution of scores for the control group, i.e., there is 0% non-overlap. An effect size of 0.8 indicates a non-overlap of 47.4% in the two distributions. An effect size of 1.7 indicates a non-overlap of 75.4% in the two distributions and that the mean of the experimental group is at the 95.5 percentile of the control group.

Calculating effect sizes

The nature of the effect size will vary from one statistical procedure to the next (e.g., it could be the difference in defects found, or a standardized mean difference, or a correlation coefficient). However, its function in power analysis is the same in all procedures. There is a wide array of formulas used to measure effect size. In fact, effect sizes can be calculated in two ways:

1. as the standardized difference between two means, or

2. as the correlation between the independent variable classification and the individual scores on the dependent variable, which is called the "effect size correlation" (Rosnow & Rosenthal, 1996)

Effect sizes: Cohen's d and Cohen's f
A posteriori power analysis

In the context of a t-test, the effect size Cohen's d was used. Cohen's *f* was used in the context of ANOVA, respectively ANCOVA. Power is the ability to detect an effect if there is one (i.e., the probability that the test will reject a false null hypothesis). Expressed as a quantity, power ranges from 0 to 1, where .95 would mean a 5% chance of failing to detect an effect that is there. In this experiment an a posteriori power analysis was conducted, which was determined by using the significance level, sample size, and calculated effect size (i.e., by assuming the effect size in the sample size is equal to the effect size of the whole population). Table 76 shows the effect sizes and the results of the power analysis. Cohen defined effect sizes as "small, d = .2," "medium, d = .5," and "large, d = .8" (Cohen, 1988). For those variables where power <.80, the required sample size has been calculated. The results will be discussed in the next section.

Table 76 Overview of effect size results and power analysis

	N	effect size d *	partial eta squared η_p^2	effect size f**	p-value	power (1-β)	Required sample size ***
ucorr (paired t-test)	18	.85	-	-	.002	.964	-
ANCOVA (F-test)	18	-	.432	.87	.002	.923	-
know_diff (paired t-test)	19	1.64	-	-	.000	1.00	-
ANCOVA (F-test)	19	-	.741	1.69	.000	1.00	-
know_diff_remember (paired t-test)	16	1.02	-	-	.001	.988	-
know_diff_understand (paired t-test)	19	1.06	-	-	.001	.997	-
know_diff_apply (paired t-test)	18	.57	-	-	.027	.751	21
ind. t-test (correcting for period effect)	9+9	1.47	-	-	.006	.911	-
know_diff_analyze (paired t-test)	19	.76	-	-	.004	.940	-
know_diff_create (paired t-test)	18	.56	-	-	.029	.737	22
aeff (paired t-test)	18	.75	-	-	.006	.918	-
ANCOVA (F-test)	18	-	.372	.77	.006	.848	-
acomp (paired t-test)	18	.57	-	-	.027	.751	21
ind. t-test (correcting for period effect)	9+9	1.47	-	-	.006	.911	-
aaccu (paired t-test)	19	.48	-	-	.049	.650	29
inf_qua (paired t-test)	19	.63	-	-	.013	.845	-

* Cohen d (input: mean, standard deviation, and correlation between groups)

** Cohen f (input: partial eta squared)

*** required sample size to achieve a power > .80

7.6 Discussion of the Analysis Results

Learning spaces improve experience reuse, knowledge acquisition, and perceived information quality

The experiment revealed that the learning spaces lead to better understanding and application of an experience package as well as to higher overall knowledge acquisition, $p < 0.05$ for all dependent variables (see Table 76 for the statistical details). Since the goal of this experiment was to provide accurate effect sizes and power values for future studies in the field, a lot of effort has been put into improving the reliability of the dependent variables and disturbing factors (e.g., only disturbing factors with a reliability > .80 were used for analysis). In addition, the usage of a crossover design may lead to confounding effects, which bias the statistical tests. Therefore, these effects have been investigated in detail; corrections have been made where necessary. A regression analysis showed that there are significant regression coefficients between the covariates and the dependent variables (see Appendix C.6). However, the application of ANCOVA did not find any significant interactions between the disturbing factors and the dependent variables. Nevertheless, in some cases, the measurement error could be reduced (see previous Table 76).

The subjects using learning spaces had a significantly higher understanding correctness than the group provided with the information of experience packages ($p = .002$, power = .923). Hypothesis $H_{0.1}$ can be rejected. A high effect size for understanding correctness ($d = .85$) was found. A

slightly higher effect size f was obtained by applying ANOVA (f = .87), which refers to a high effect (Cohen, 1988).

Medium to large effects for knowledge acquisition

A significant effect has been found for all knowledge acquisition variables. The highest effect was found for the overall knowledge acquisition difference (f = 1.69, power = 1.00). Higher effects were detected for the lower cognitive levels remember (d = 1.02, power = .998) and understand (d = 1.06, power = .997). Medium effect sizes were obtained for the levels apply (d = .57, power = .751), analyze (d = .76, power = .940), and create (d = .56, power = .737). The power of the hypothesis test for $H_{0.2.2.apply}$ is only .737, which means that there is still a 26.3% chance that the effect is not detected, i.e., that $H_{1.2.2.apply}$ cannot be accepted because a minimum of .80 was required for power in Section 7.2. $H_{1.2.2.apply}$ could probably have been accepted with a slightly larger sample size (we would have needed 21 subjects for both groups). Due to the fact that a significant period effect was detected for the level of apply (p = .006), a correction for period effect was done, which resulted in a lower significance (p = .006, power = .911). Furthermore, $H_{1.2.2.create}$ cannot be accepted because of the low power (power = .739). $H_{1.2.2.create}$ could probably have been accepted with a slightly larger sample size (we would have needed 22 subjects for both groups). To sum up, all null hypotheses related to knowledge acquisition difference can be rejected but the alternative hypothesis $H_{1.2.2.create}$ related to the level create cannot be accepted.

Medium effects for application

Regarding the application variables, medium effects have been detected. However, after correcting a period effect the effect, size for application completeness increased up to d = 1.47. By applying an ANCOVA, the measurement error was slightly reduced by the disturbing factor experience in software quality assurance (exp_sqa), even though the interaction between exp_sqa and application efficiency was not significant (p = .153). The power was reduced from .918 to .848 due to a reduction in the degree of freedom by one when a covariate is added to the model. The alternative hypothesis for application accuracy could not be accepted because the power was .650. A sample size of 29 could probably lead to a power higher than .800. Hence, all null hypotheses can be rejected; however, the effect measured for application accuracy cannot be detected with an acceptable probability, which means that $H_{1.3.3}$ cannot be accepted.

No significant correlation could be found between the understanding variable, the application variables, and the overall knowledge acquisition difference variables. Nevertheless, a medium positive correlation coefficient was found between understanding correctness and knowledge acquisition difference (r = .315, p = .203 based on a two tailed Pearson correlation test). Similar correlation coefficients were found between understanding correctness and application efficiency (r = .399, p =.219)

and understanding correctness and application accuracy (r = .344, p = .162) but they were not significant.

No significant correlations were found between the lower level of cognition (i.e., remember and understand) and the application variables. An interesting fact was that for higher cognitive levels, the correlation coefficients were slightly higher and one significant correlation was detected between knowledge acquisition difference apply and application efficiency (r = .642, p = .005).

Medium effects for perceived information quality

Finally, the perceived information quality of the provided learning spaces was significantly higher than that of the experience packages (p = .013, power = .845). Hence, the null hypothesis $H_{0.4}$ can be rejected and the alternative hypothesis $H_{1.4}$ can be accepted.

The experiment provides valuable results for future studies

In the past, no similar evaluations have been done. Hence, a comparison with statistical results from previous work is not possible. Nonetheless, this experiment provides valuable results for future studies. Effect sizes can be used to calculated sample sizes of future evaluations and provide a basis for comparisons. Section 9.2 will suggest a number of directions that can be taken by future research.

7.7 Threats to Validity

In the following, different types of threats to validity will be discussed. Results have adequate validity if they are valid to the population to which we would like to generalize (Wohlin et al., 1999). Cook and Campbell distinguish between conclusion validity, construct validity, internal validity, and external validity (Cook & Campbell, 1979).

7.7.1 Conclusion Validity

Conclusion validity is also called statistical conclusion validity and is the degree to which statistically significant conclusions can be drawn from the data output of the experiment. The literature study about confounding effects and the results of their investigation led to several corrections of these effects, which made the conclusions more reliable. The selection of tests was made by checking their required assumptions in detail (e.g., normality test or checking the assumptions for applying ANCOVA). Moreover, conclusions in this experiment were made based on the results of significance testing with a significance level of .05 and a minimum required power of .80, which is challenging for an explorative study like this one. Nevertheless, all hypotheses have been rejected with p-values < .05. The validity of an experiment depends on the reliability of the disturbing and dependent variables and of course on the related measures. Several measures were taken to enhance the reliability of the variables (e.g., principal component analysis of the experience level vari-

ables, item analysis of the knowledge acquisition questionnaire, etc.). In addition, a pilot test was performed with two students to discover bad questions, respectively poor wording of questions. The experimental setting was kept constant, e.g., no noise, no interruptions caused by technical problems, no different environment for the two days, etc. Regarding random heterogeneity of the subjects, the subjects were assigned randomly to the groups based on four levels of experience (i.e., their experience level was determined by evaluating the briefing questionnaire).

7.7.2 Construct Validity

Construct validity is the degree to which the independent and dependent variables accurately measure the concept they intend to measure (i.e., the treatment reflects the cause effect and the experiment outcome reflects the effect construct) (Cook & Campbell, 1979).

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 factual and conceptual as well as procedural knowledge was assessed. The different questions and assignments refer to cognitive activities according to the well-recognized learning objectives taxonomy of Anderson and Krathwohl (L. W. Anderson & Krathwohl, 2001). The item analysis ensured that only items with a strong contribution to the overall knowledge acquisition measuring constructs were used for analysis. The constructs for experience levels had been checked with a principal component analysis. Several items had been deleted due to multiple ambiguous factor loadings on several experience variables.

The difficulty with the measurement of completeness, efficiency, and accuracy was to define which code smells are real defects, and which are not. A refactoring expert checked all the assignments and questionnaires to evaluate the proposed sample solutions that were used to get the number of correct defects found, respectively to decide which proposed refactorings were correct. In addition, a one-day workshop was done after the experiment to discuss the experiment and the proposed refactoring together with the subjects and the refactoring expert. By doing this, it could be confirmed that the learning space led to a very similar understanding of refactoring and hence identification and removal of code smells as the expert's.

Regarding use, acceptance, and software ergonomics (see Chapter 8), well-defined measure instruments were used. In general, none of the disturbing factors or dependent variables was assessed by one single measure – several measures were used to adequately measure the constructs.

7.7.3 Internal Validity

Internal validity is related to the fact that if there exists a relationship between the factors and the dependent variables, we must be sure that it is a causal relationship, and not a result of a factor that has not been considered.

Since the experiment took place on two subsequent days, some of the improvements related to a treatment may be attributed to other happenings that took place between the two periods. However, the subjects were told not to discuss the experiment or do things related to refactoring (e.g., read about refactoring before the second day). Maturation is a threat related to processes taking place within the subjects (e.g., becoming tired or learning new things). Nevertheless, a detailed analysis of confounding effects and the short duration of the experimental tasks should either detect such effects or simply reduce them sufficiently. Another relevant threat is testing, i.e., the subject may respond differently when a test is repeated. Since the knowledge acquisition test was used before and after an experiment period, this threat must be considered. Nevertheless, the subjects were not provided with any feedback after an experimental period. Since only a conventional Web browser was used, there was no risk that the subjects underwent a maturing effect regarding the tools used during the experiment. In addition, the material used during the experimental periods was completely different, i.e., different code smells, Java code, related experience packages, and learning spaces were used during the two days. The instrumentation threat is not relevant here because the material had been tested in a small pilot study and no problems had been detected. Hence, the type and form of the used material should not have an impact on the treatment effect. In addition, no complaints about the material were made after the experiment. No significant difference was found regarding the complexity of the information provided during the two days (this variable was assessed with the debriefing questionnaire; $p = .759$). The distribution of scores for the questions, respectively assignments and sample solutions, was checked by a refactoring expert. An attempt was made to minimize any selection effect by randomly assigning the subjects. ANCOVA was applied to reduce the between group variance. Furthermore, the effect of letting volunteers take part in an experiment may influence the results. Volunteers are generally more motivated and suited for a new task than the whole population. However, in this experiment, the subjects had to participate in the experiment because refactoring was a mandatory part of their practicum.

7.7.4 External Validity

External validity is the degree to which the outcomes of the experiment can be generalized to the whole population.

One problem 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. 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. The Java software system used in this experiment was developed by 19 students during 15 full-time weeks, and consists of 96 classes. This corresponds to a possible small application in industry, which again means that similar effects can be expected when the experiment is replicated in an industrial context.

8 Use and Acceptance Evaluation

"You cannot have a science without measurement" (R. W. Hamming)

The learning space approach has been evaluated twice regarding its use and acceptance: first, at the end of the experiment presented before and second, in more detail, during a case study with a primarily focus on evaluating use, acceptance, and software ergonomics. The statistical tests refer to hypothesis $H_{1.5}$, which test whether the use, acceptance and software ergonomics of the learning space user interface were significantly positive in an experiment (Section 8.1.1) and a case study (Section 8.1.2):

- All investigated use and acceptance factors were significantly positive with a p-value lower than 0.05.
- The results of the case study with experience management and technology-enhanced learning experts where even better than the results of the experiment.

8.1.1 Use and Acceptance Evaluation (in the Context of the Experiment)

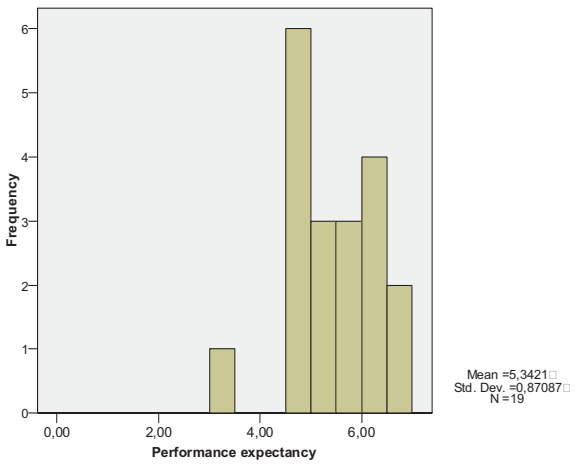
In order to assess the use and acceptance of the learning space approach, the subjects were asked to fill out a questionnaire, which was part of the debriefing questionnaire after the two experimental periods (see Appendix B for the details about the questionnaire). The questions were based on the model of Unified Theory of Acceptance and Use of Technology (UTAUT) (Venkatesh, Morris, Davis, & Davis, 2003), which is a measurement instrument based on the former measurement instrument of the Technology Acceptance Model (TAM) (Davis, 1986, 1989). UTAUT contains constructs for measuring performance expectancy, effort expectancy, attitude towards using technology, intention, anxiety, self-efficacy, facilitating conditions, and social influence. For this experiment only performance expectancy, effort expectancy, and attitude towards using technology were gathered because the others could not be answered by the subjects in the context of the experiment. *Performance expectancy* is defined as the degree to which an individual believes that using the system will help him or her to attain gains in job performance. *Effort expectancy* is defined as the degree of ease associated with the use of the system. *Attitude towards using technology* is defined as an individual's overall affective reaction to using a system (Venkatesh et al., 2003).

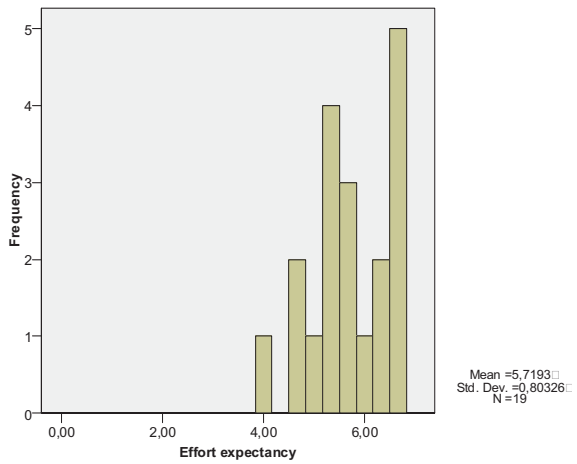
Each construct contains several question items with a 7-point Likert scale. The factors have been calculated by summing up the values of the question items divided by the total number of questions (i.e., "1" corresponds to "strongly disagree", …, "7" corresponds to "strongly agree"; range of a factor =[1,7]).

For all three factors, the median is higher than the value of four, which is the neutral value of the scale. In addition, the standard deviations were quite low, which corresponds to small variance between the subjects. Performance expectancy was rated the lowest of all. The related question items tap into an individual's liking, enjoyment, joy, and pleasure associated with technology use. The data was gathered after two days of experimentation. One reason for this could be that enjoyment, etc. were influenced by the duration and the difficult tasks of the experiment itself. Nevertheless, a higher performance and effort expectancy was received. A Shapiro Wilk test confirms that the distributions are not significantly different from a normal distribution. The following table shows the results of a one-sample t-test testing whether the results are significantly different from the neutral value of 4.

Table 77 Descriptive statistics of UTAUT factors for learning spaces (experiment)

UTAUT factors	N	Min.	Max.	Mean	Median	Std. Dev.	Skewness	Kurtosis
Performance Expectancy	19	3.25	7.00	5.34	5.25	.87	-.284	.603
Effort Expectancy	19	4.00	6.67	5.72	5.66	.80	-.393	-.615
Attitude towards using technology	19	3.50	6.75	4.92	4.75	.85	.341	.188





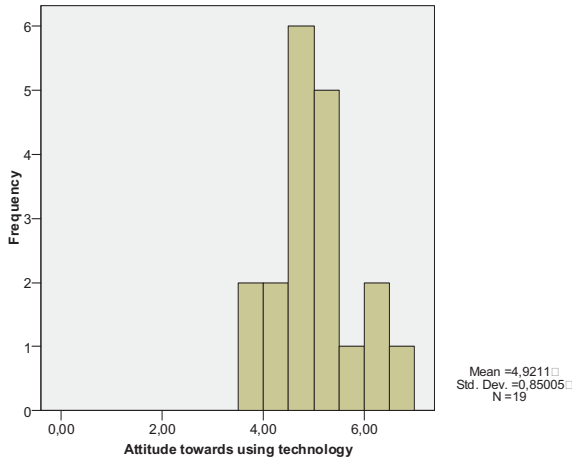


Figure 51 Histograms of UTAUT factors for learning spaces (experiment)

Highly significant results were obtained (p = .000) for all three factors. In addition to the quantitative data, the subjects provided qualitative feedback that was used for improvements to the system in the future. Most of this feedback was related to user interface issues.

Table 78 One-sample t-test for UTAUT factors for learning spaces

	Test value = 4						
	Mean Difference	95% Confidence Interval of the Difference		df	t	Crit. $T_{0..95}$	p-value
		lower	upper				
Performance expectancy	1.34	.92	1.76	18	6.718	1.734	.000
Effort expectancy	1.72	1.33	2.11	18	9.330	1.734	.000
Attitude towards using technology	.92	.51	1.33	18	4.723	1.734	.000

8.1.2 Use and Acceptance Evaluation (Case Study)

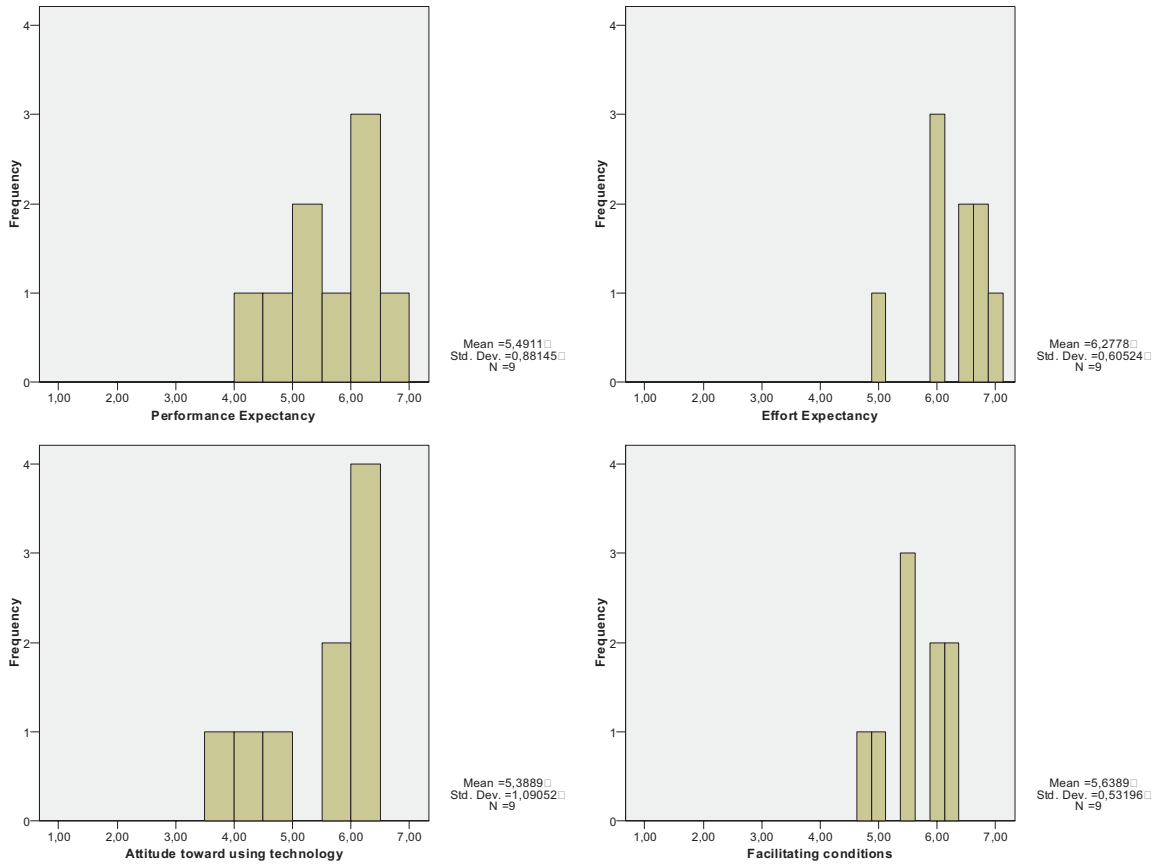
Half a year later, another more detailed use and acceptance evaluation was done with 10 researchers of Fraunhofer IESE. Since the experiment, smaller improvements had been made to the user interface of learning spaces. However, the access to a learning space, the information provided by the learning spaces, and the experience packages were exactly the same as during the experiment. All the researchers had a stronger background in experience management and educational systems than the subjects who took part in the experiment. During 30 minutes, the subjects received an introduction to the learning space approach in general and the system in particular.

Two additional factors were measured: *Facilitating conditions* are defined as the degree to which an individual believes that an organizational and technical infrastructure exists to support use of the system. *Self-efficacy* is related to the judgment of one's ability to use a technology (e.g., a computer) to accomplish a particular job or task (Venkatesh et al., 2003). The descriptive results are presented next.

Table 79 Descriptive statistics of UTAUT factors for learning spaces (case study)

UTAUT Factors	N	Min.	Max.	Mean	Median	Std. Dev.	Skewness	Kurtosis
Performance Expectancy	9	4.00	6.75	5.49	5.67	.88	-.404	-.601
Effort Expectancy	9	5.00	7.00	6.28	6.50	.60	-1.117	1.537
Attitude towards using technology	9	3.50	6.50	5.39	5.50	1.09	-.752	-.716
Facilitating conditions	9	4.75	6.25	5.64	5.50	.53	-.449	-.875
Self-efficacy	8	4.00	7.00	5.81	6.12	1.17	-1.067	-.408

Looking at the results, it seems that use and acceptance were rated even higher than in the experiment. The median exceeds the value of 5.50 for all factors. A one-sample t-test compares the means with the neutral value of 4 and provided p-values lower than 0.005 for all factors. However, the qualitative feedback showed that there is still potential to improve the attitude towards using the technology and self-efficacy. A help system and better presentation of the learning space content (e.g., barrier-free presentation of information) could further improve the use and acceptance of the learning space approach.



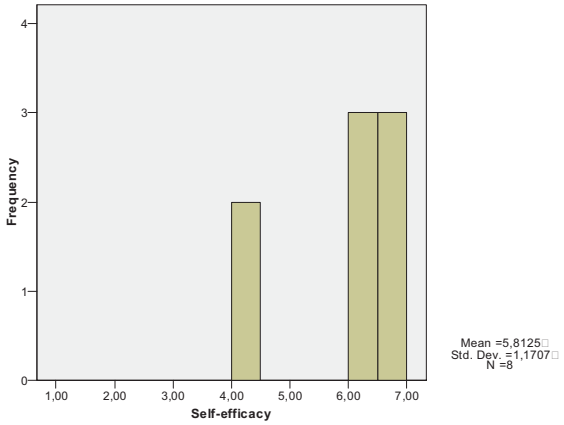


Figure 52 Histograms of UTAUT factors for learning spaces (case study)

In addition to UTAUT, a German questionnaire for measuring software ergonomics was used. The questionnaire used was conformant to ISONORM 9241/110 and can be used for evaluating software that has already been used for a longer time or for prototypes (International Organization of Standardization, 2008) (ergo-online, 2008). "Aufgabenangemessenheit" measures whether the system helps the user to solve his problems without burdening him. "Selbstbeschreibungsfähigkeit" refers to the system's ability to describe its functionalities and its understandability. "Steuerbarkeit" is defined as the possibility the user has to influence the way of working with the system. "Erwartungskonformität" refers to how good a system fulfills the expectations and habits of the user. "Individualisierbarkeit" is defined as the system's ability to be adapted to the user's personal needs. "Lernförderlichkeit" refers to how easy it is to become familiar with the system and how the system supports the user in learning new functionalities. The construct of "Fehlertoleranz" was not suitable for this context, and was therefore not considered. The same scales were used as during the experiment.

Table 80 Descriptive statistics of ISONORM factors for learning spaces (case study)

ISONORM Factors	N	Min.	Max.	Mean	Median	Std. Dev.	Skewness	Kurtosis
Aufgabenangemessenheit	10	5.00	6.50	6.00	6.12	.47	-1.119	.978
Selbstbeschreibungsfähigkeit	10	5.00	6.33	5.87	6.00	.52	-.877	-.544
Steuerbarkeit	10	4.80	7.00	5.97	6.20	.76	-.416	-1.273
Erwartungskonformität	10	5.80	7.00	6.32	6.20	.38	.600	-.468
Individualisierbarkeit	9	1.50	6.25	4.44	4.50	1.50	-.807	.444
Lernförderlichkeit	10	4.60	6.80	5.66	5.70	.75	-.120	-.970

The median of almost all factors exceeds the value of 6.00, which is very high. Only for "Individualisierbarkeit", the value was much lower (*MD* = 4.50). The qualitative feedback of the open face-to-face interviews con-

ducted with the subjects after filling out the questionnaires showed that the reason for this low score were wrong expectations of the system. They were told they would be introduced to a system capable of providing information in a user-sensitive way. However, during this case study, the adaptivity (i.e., user model) was not activated because the focus of the case study was on the learning approach and content presentation rather than on the aspects of adaptivity and personal individualization. A one-sample t-test compared the means with the neutral value of 4 and provided p-values equal to 0.00 for all factors but "Individualisierbarkeit" (p = .401).

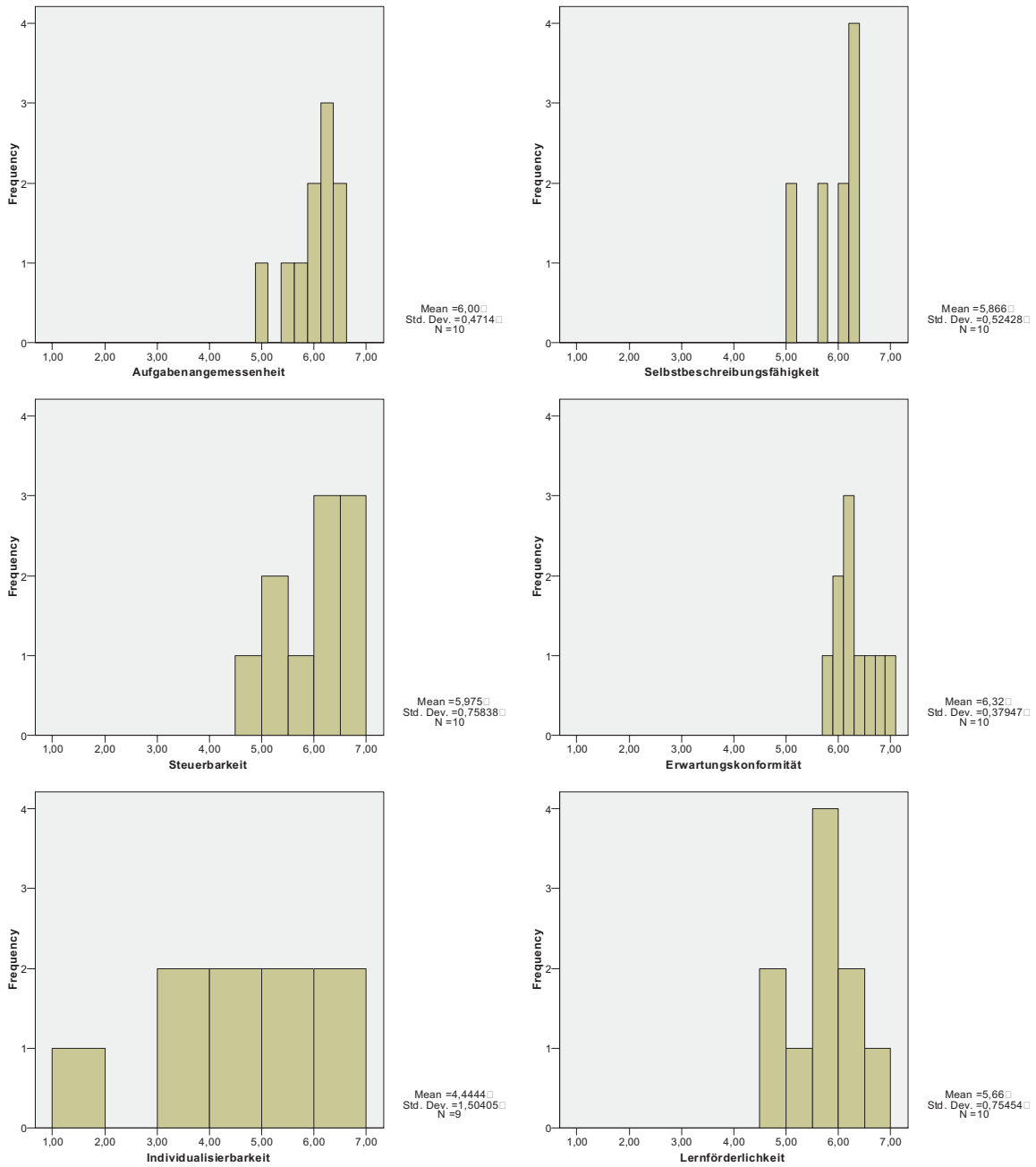


Figure 53 Histograms of ISONORM factors (case study)

9 Summary and Outlook

"The only source of knowledge is experience"
Albert Einstein

This chapter summarizes the main contributions and results of this thesis (Section 9.1). In addition, it lists the limitations of this thesis and provides a research agenda for future work (Section 9.2), and finally states some concluding remarks (Section 9.3).

9.1 Results and Contributions

Problems addressed

The main achievement of this work lies in the design, implementation, and validation of the learning space approach in order to address three central problems in practice (see Section 1 and Section 2):

- Bad understanding of reusable artifacts in general and experience packages in particular
- No explicit support for internalization of knowledge and no compliance with human information processing
- No explicit connection between KM/EM and technology-enhanced learning approaches

Solution: learning spaces

Learning spaces intend to improve the understanding and application of experience packages by restructuring information contained in the experience package description in a way that learning processes are stimulated and knowledge acquisition is fostered. In addition, they improve the perceived information quality. The learning space approach offers contributions to the current state of the art in software engineering as well as knowledge management, technology-enhanced learning at the workplace, and adaptive hypermedia approaches in particular.

Extensions to the reuse model

From the perspective of the reuse model, three extensions were necessary to enable context-aware adaptation and generation of learning spaces (see also next figure):

- *Object Interface* (Dependencies) was extended by *Related Domain Concepts* CC_i
- *Object Context* (Solution Domain) was extended by *Related Context Concepts* DC_i
- *Activity – Mechanisms* was extended to characterize the generation and adaptation activity (i.e., *General Adaptation, Adaptation Type*

Adaptation Level, Adaptation Navigation Techniques, Adaptation Presentation Techniques)

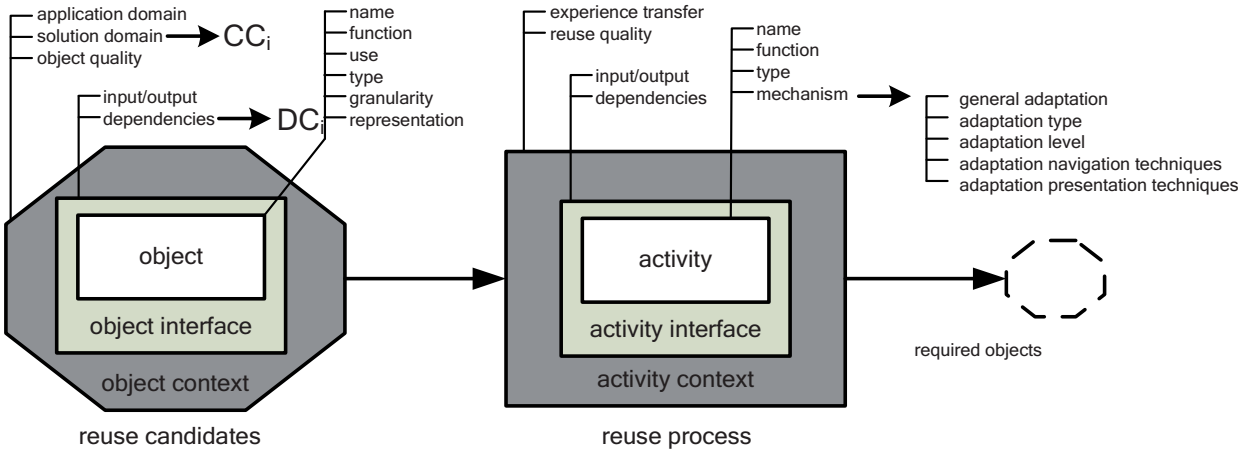


Figure 54 Reuse model of Basili and Rombach (V. R. Basili & Rombach, 1991)

Three extensions to the experience factory infrastructure

From the perspective of the experience factory, the learning space approach extends the "project support" activity (see Figure 55). It reuses information from the software organization platform database (SOP DB) such as situational content describing situations in projects, learning content, and experience packages. This information is then merged into a learning space; variabilities of the learning space are resolved based on context and domain characteristics; the generic artifacts of a learning space are adapted, and, finally, presented to the user in the project.

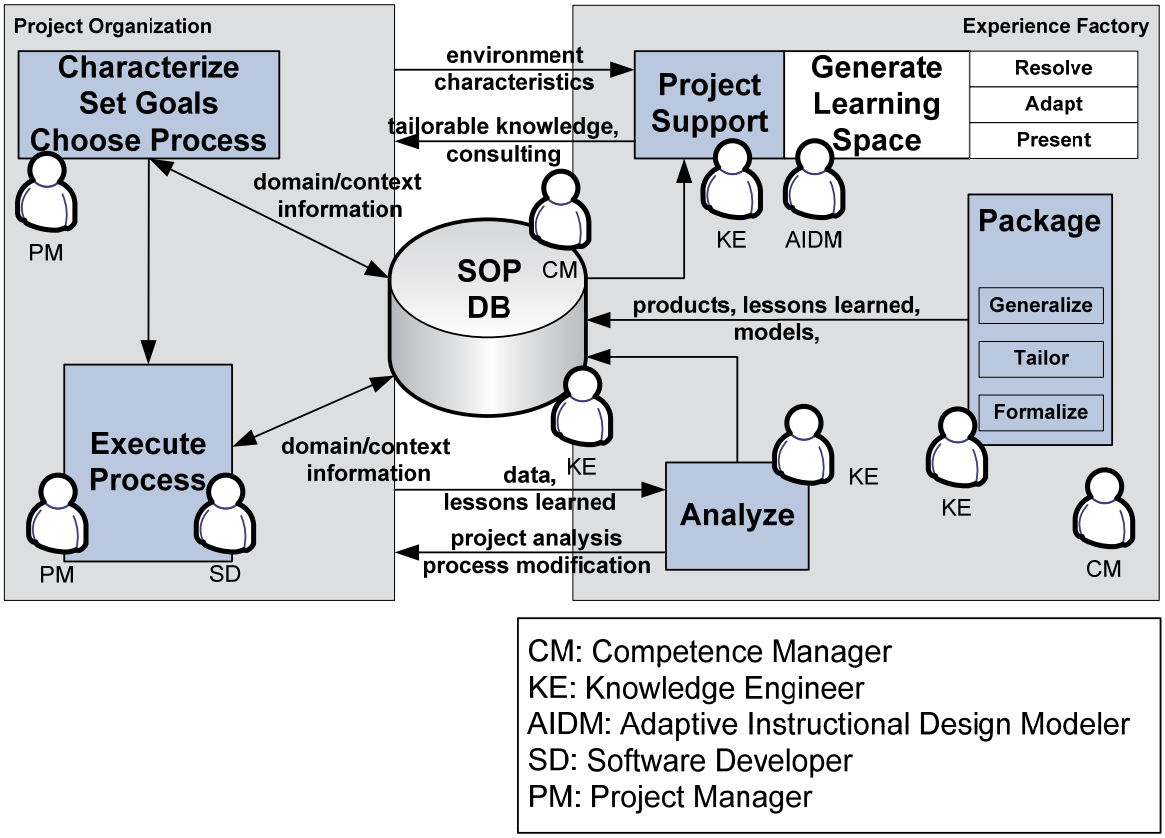


Figure 55 Extension of the experience factory

Second, the experience base is not only part of the experience factory, but it is now a storage medium between the project organization and the experience factory: Different types of content are created, stored, and reused from both sides - users have become both content producers and consumers, which supports a more open knowledge sharing community between the project organization and the experience factory. Third, the role model of the EF has been extended by the adaptive instructional design modeler, who is responsible for assigning learning goals and learning methods to the identified learning scenarios and the target group. He designs the structure of the learning space, identifies possible variants and specifies them explicitly in the variability model, respectively the decision model.

The main contributions are classified into theoretical, practical, and empirical work.

I. Theoretical work

The contributions to theory refer to the identification of problems by first performing an extensive literature survey in the domain of software reuse in general and KM/EM in particular. The outcomes were used to perform two case studies and one market survey in order to confirm the identified problems and derive a first set of requirements for the learning space approach.

An overview of the state of the art in the domain of professional acting and experiential learning was done to get a basic understanding of the ongoing learning processes that should be stimulated by the learning space approach in order to support experiential learning.

Context model and domain model

The classification of KM/EM approaches helped to identify possible information types that can be used to describe situations in software engineering. The outcome was used to define the context model, which serves to describe situations in software engineering by means of different context concepts and relationships. In addition to the context model, a domain model was developed to model the body of knowledge in software engineering by means of different domain concepts and relationships. This domain model is used to annotate experience packages and instructional content.

Learning space model

The developed learning space model defines learning spaces on different levels of abstraction (i.e., structure, content, and presentation) through an instructional design model and a learning resource model. It separates structure, content, and presentation of a learning space and supports the adaptation of its generic artifacts based on the variability model.

Variability model

The learning space approach is able to automatically generate context-aware learning spaces. This requires that the learning space can be adapted to the actual context at development time (before learning) and at runtime (during learning). A variability model allows defining variabili-

ties on different level of abstraction and supports the static and dynamic resolution of the variation points. Variation points realize variabilities and are part of a generic artifact. They describe the location in a generic artifact where the adaptation will occur.

Related techniques

All the models, including their concepts and relationships have been defined in predicate logic. These models were used by several techniques for the systematic and automatic, on-demand generation of learning spaces. They are related to the activities of resolving, static and dynamic adaptation, and presentation of generic artifacts.

Lifecycle model

A lifecycle model illustrates the different states of a learning space and explains when a learning space is generated, adapted, and presented. It helps to understand the differences between static adaptation by means of static context indicators and dynamic adaptation by means of dynamic context indicators.

Role model

A role model was defined to describe the different types of activities and responsibilities in the context of the extended experience factory. The model explains the difference to existing role models in the domain of experience management and helps to implement the learning space approach in software engineering.

Selecting learning strategies and methods

A critical part of the learning space approach is the selection of the right learning strategies and learning methods for supporting a specific learning scenario. In this work, appropriate strategies and methods have been selected to stimulate experiential learning and to foster knowledge acquisition during experience package reuse.

II. Practical work

The contributions to practice are related to the implementation of the different models, methods, techniques, and tools.

Implementation of the models

The context model was developed by using the Wiki syntax for software engineering situations. The domain model was described by a domain ontology in the OWL language. Several DTD schemas for decision model, resolve model, and instructional design templates have been developed.

Implementation of the learning space approach

The learning space approach has been implemented as a plugin in the software organization platform, including tools for describing and annotating learning content (i.e., a vocabulary manager and an authoring tool for learning elements). This supports the conceptual as well as technical integration of knowledge/experience management and technology-enhanced learning. In addition, a lightweight experience management system for documenting and retrieving experience packages was developed.

III. Empirical work

The empirical evaluations provide statistical significant results that quantify the impact of learning spaces upon the understanding and applica-

tion or experience packages, upon knowledge acquisition, and upon perceived information quality. A power analysis and effect sizes provide a strong baseline for future evaluations and meta-analysis studies.

Experiment design and analysis

Due to the fact that experiments related to the didactical augmentation of software engineering experience packages have not been conducted before, this experiment serves as an exploratory evaluation, which can be used as a baseline for future evaluations and developments in this area. Therefore, strong emphasis has been put upon the construction of reliable measurement instruments (by applying a principle component analysis and item analysis), the selection of suitable disturbing factors for controlling the experiment, and the data analysis. Particular importance is put on the analysis of confounding effects. A literature survey was done to clarify the terminology and identify the different statistical approaches for their identification and correction. The reason for this survey was that these effects are either mostly neglected in empirical software engineering or examined incorrectly.

Summary of the results: statistical hypotheses

The statistical data analysis revealed that all eleven null hypotheses related to understanding, application, knowledge acquisition variables, and perceived usefulness could be rejected, which means that learning spaces have a high potential of improving experience package reuse. The subjects using learning spaces had a significantly higher understanding correctness than the group provided with the information from experience packages ($p = .002$, power = .923). Regarding knowledge acquisition, higher effects have been detected for the lower cognitive levels remember ($d = 1.02$, power = .998) and understand ($d = 1.06$, power = .997). Medium effect sizes were obtained for the levels apply ($d = .57$, power = .751), analyze ($d = .76$, power = .940), and create ($d = .56$, power = .737). Finally, regarding application efficiency ($f = .77$, power = .848), application completeness ($d = 1.47$, power = .911), and application accuracy ($d = .48$, power = .650), medium, respectively high effects (Cohen, 1988) have been detected. Finally, the perceived information quality of the provided learning spaces was significantly higher than that of the experience packages ($d = .63$, power = .845).

Summary of the results: research hypotheses

Regarding the research hypotheses stated in Section 1.4 the expected improvement goals could not be met for understanding correctness (20%), which refines the variable understanding. Furthermore, the improvement goal could also not be fulfilled for perceived usefulness (17%). Both were expected to be at least 25%. The reason for this could be that the templates used for describing the experience packages were already of good quality – they correspond to state-of-the-practice experience packages. This was necessary in order to create a fair comparison situation between conventional experience packages and experience packages enriched by a learning space. Nevertheless, the improvement goal for knowledge acquisition have been fulfilled to a large extent and were much higher than expected (i.e., the cognitive level of apply with

80%, up to 275% for the cognitive level of understanding). Application efficiency (53%) and application completeness (51%), which refine the application variable, also exceeded the expected improvement rate of 25%.

Use, acceptance, and software enconomics

The experiment and an additional case study showed that good or very good results were achieved with the learning space approach regarding technology use and acceptance, and software ergonomics.

9.2 Limitations and Future Research

Despite the good results of the learning space evaluation, limitations of the approach still exist, which provides room for future research. Since the development of the learning space approach also requires the consideration of different disciplines, the derived research fields are also interdisciplinary: software engineering and software product lines in particular, artificial intelligence in education, learning theories and methods, and empirical research.

Limitation: Generation of learning spaces is not based on distance measures between the selected reuse candidate and the requirements

The standard reuse process consists of *identifying* reuse candidates from the reuse repository, *evaluating* the reuse candidates and *selecting* a candidate, *modifying* the candidate before reuse of necessary, and *integrating* or applying the experience (V. R. Basili & Rombach, 1991). By adding an adaptation and generation activity to this reuse process, the software engineer is supported in bridging the gap between reuse candidate and required object. However, the learning space is not generated and adapted based on a precise distance measure between the reuse candidate and the required objects - it is done based on available context information (i.e., concepts and semantic relationships) of the experience package and the current situation.

In the future, research work should focus on exactly determining the distance between the context of the experience package and the actual working situation in order to enhance the generation of learning spaces. This requires the specification of distance measures for the different context characteristics.

Limitation: adaptive instructional design modeller

The role model introduced in this work added the role of the adaptive instructional design modeler to the standard role model for the experience factory. Beside the well-known challenges for setting up an experience factory in an organization, finding a person who is capable of performing the related activities of the adaptive instructional design modeler will be difficult: Besides pedagogical knowledge about learning theories and methods, the instructional designer needs competencies in the domain of interest and in technically specifying instructional designs for different learning methods, and must be able to reflect about all the different variants of a learning space while keeping in mind the impact-

ing contextual indicators. It is obvious that finding one person that covers all these competencies will be difficult in a software organization, since domain experts are mostly not instructional design experts and instructional designers are seldom found in industry, or their knowledge is too tacit to instantiate it in the context of technology-enhanced learning.

One possibility is to make instructional design more applicable. Initial instructional design patterns have to be developed so that a software engineering domain expert can apply them. These patterns are very useful for describing instructional design strategies in a comprehensive manner. Goodyear presents design patterns and pattern languages for networked learning (Goodyear, 2005). His patterns use a template similar to the one used by Alexander in the architectural domain (C. Alexander, 1979) and also in the software development domain (Gamma, 1995).

Limitation: Getting the critical mass of content

Educational approaches such as the learning space approach only work when a critical mass of situational and learning content is available for building the learning spaces. Many intelligent technology-enhanced learning approaches have struggled in the past because of a lack of content chunks in their repositories.

In order to cope with this problem, content from open repositories of content such as Wikipedia or other learning object repositories should be integrated into the learning spaces. Even content from Web2.0 platforms should be considered for future applications. Technologies such as mashups, which integrate information from different Web2.0 technologies, will impact research on and development of approaches such as the learning space approach in the future.

Information agents are a special kind of intelligent software agents (Wooldridge & Jennings, 1995). They could provide a solution by means of retrieving, analyzing, manipulating, and fusing heterogeneous information, as well as visualizing and guiding the user through the available individual space. Agent technology made its way to education about 10 years ago in the domain of *Artificial Intelligence in Education*.

Limitation: Tool support

The development of several models requires tool support. Regarding the domain ontology, Protégé was used, which is a comprehensive tool for modeling ontologies. However, regarding the decision model, resolution constraints, etc. a conventional XML editor was used, which makes it almost impossible to develop complex decision models and to keep the resolution constraints consistent. Here, recently developed tools from the domain of product line engineering could be adapted for the purpose of learning space adaptation since the available tools from the adaptive hypermedia domain are to proprietary to be used in practice. They need to support the modeling of variabilities across different abstraction layers and they must provide different, user-friendly views on decision models so that inconsistencies in the model can be detected easily.

Limitation: Empirical research

The threats to validity indicate that the experiment should be replicated in a domain other than refactoring to confirm the results. Additional investigations in industry would strengthen the findings even if the experiment's subjects were comparable to software developers in practice. Other research opportunities would be to evaluate the approach with decision makers (e.g., project managers); to predict the effect of learning spaces upon knowledge acquisition in order to build predictive models for learning spaces; and to compare different adaptation techniques.

In addition, the statistical analysis has shown that confounding effects in crossover studies (i.e., within-subject/repeated measure designs) are still being neglected. It seems that no perfect approaches exist to detect or correct them. Because repeated measure designs are very common in software engineering, research should be done to analyze statistical approaches from other fields (e.g., clinical research) and to investigate them in the domain of software engineering.

Limitation: Learning theories and methods

The existing world of education is currently experiencing rapid changes with to all the new technologies and methods coming up on the market. This change is also taking place in technological and in instructional methods used in traditional as well as in technology-enhanced learning. Current learning theories and methods should adapt to the current developments in the information society. George Siemens, for example developed the learning theory of "connectivism" (Siemens, 2004). He states that "the people earn their knowledge from forming connections to specialized information sets, and the connections that enable us to learn more are more important than our current state of knowing". This matches with the recent developments in the Web2.0 and Web3.0 areas. In software engineering, these changes in knowledge creation and sharing will also significantly impact the way students should be educated and how software engineers develop software in practice. In 2008, the community of software engineering education invented the term "Net Generation of Software Engineers" – people who have never lived without the Internet and who develop software in an open and distributed environment.

Research agenda

The following research questions can be derived:

- What are the most common learning scenarios in software engineering at the workplace and which instructional design pattern set can implement these scenarios so that short-term problem solving as well as long-term competence development can be supported?
- How can the exact distance be determined between the context characteristics of the selected candidate and the actual context of the required object?
- Can the problem of critical mass of content be addressed by integrating open content repositories for software engineering, and how can

intelligent agents support the retrieval of potential learning content for learning spaces in software engineering?

- How can available techniques and tools from the domain of product line engineering be adapted and used for describing the variabilities in a learning space?
- What types of confounding effects impact repeated measure designs in software engineering, and how can we correct them or even prevent them?
- Which replications of the experiment are necessary for building predictive models of knowledge acquisition for learning spaces in software engineering?
- What are the most important static and dynamic indicators for context-aware adaptation, and what are the most effective adaptation techniques for learning spaces in software engineering?
- What are the characteristics of the upcoming net generation of software engineers, and how should information-based services be adapted so that they address their information needs and educational goals?

9.3 Concluding Remarks

The learning space approach has shown that experience management is not only a matter of building up an experience factory in an organization, convincing management at its usefulness, developing new schemata for describing experience, or improving such things as search algorithms. As long as reuse-based approaches do not structure information compliant to human information processing, their success will be short-lived due to the fact that the processes of constructing new knowledge and using it in new situations are not explicitly supported. Nevertheless, the challenge is not to try to perfectly adapt the learning space to a specific user's characteristics and his situations in order to support learning processes – it is a matter of reducing the learning options and producing the right set of possible learning paths, so that the learner can choose on his own how to reach the intended goal.

Abbreviations

AHS	Adaptive Hypermedia System
EAHS	Educational Adaptive Hypermedia System
EM	Experience Management
EMS	Experience Management System
EP	Experience Package
IMS LD	IMS Learning Design
KM	Knowledge Management
KMS	Knowledge Management System
LOM	Learning Object Metadata
SCORM	Sharable Content Object Reference Model
SOP	Software Organization Platform
TEL	Technology-Enhanced Learning

References

Abeyasekera, S., & Curnow, R. N. (1984). The Desirability of Adjusting for Residual Effects in a Crossover Design. *Biometrics, 40*, 1071-1078.

Adams, M. J. (1989). Thinking Skills Curricula: Their Promise and Progress. *Educational Psychologist, 24*, 25-77.

Akhras, F. N., & A., S. J. (2000). System Intelligence in Constructivist Learning (in SE). *International Journal of Artificial Intelligence in Education, 11*, 344-376.

Akhras, F. N., & Self, J. A. (2000). System Intelligence in Constructivist Learning. *International Journal of Artificial Intelligence in Education, 11*, 344-376.

Alavi, M., & Leidner, D. E. (1999). Knowledge Management Systems: Issues, Challenges, and Benefits. *Communications of the AIS, 1*(7).

Alexander, C. (1979). *The timeless way of building*. New York: Oxford University Press.

Alexander, P. A., Schallert, D. L., & Hare, V. C. (1991). Coming to Terms: How Researchers in Learning and Literacy Talk about Knowledge. *Review of Educational Research, 61*(3), 315-343.

Althoff, K.-D. (2001). Case-based Reasoning. In S.-K. Chang (Ed.), *Handbook of Software Engineering & Knowledge Engineering* (Vol. 1, pp. 549-587). Singapore: World Scientific.

Althoff, K.-D., Birk, A., Hartkopf, S., Müller, W., Nick, M., Surmann, D., et al. (1999). *Systematic Population, Utilization, and Maintenance of a Repository for Comprehensive Reuse.* Paper presented at the 11th International Conference on Software Engineering and Knowledge Engineering, Learning Software Organizations, Methodology and Applications, Kaiserslautern, Germany.

Althoff, K.-D., & Pfahl, D. (2003). Making Software Engineering Competence Development Sustained through Systematic Experience Management. In A. Aybüke, R. Jeffrey, C. Wohlin & M. Handzic (Eds.), *Managing Software Engineering Knowledge*. Heidelberg / New York: Springer.

Anderson, J. R. (1993). *Rules of the mind*. Hillsdale, N.J.: L. Erlbaum Associates.

Anderson, J. R., & Graf, R. (2001). *Kognitive Psychologie* (3. Aufl. ed.). Heidelberg u.a.: Spektrum Akad. Verl.

Anderson, L. W., & Krathwohl, D. R. (2001). *A Taxonomy for Learning, Teaching, and Assessing: a Revision of Bloom's Taxonomy of Educational Objectives* (Complete ed.). New York: Longman.

Arango, G., Baxter, I., Freeman, P., & Pidgean, C. (1985). *Maintenance and Porting of Software by Design Recovery.* Paper presented at the Conference on Software Maintenance, Los Alamitos, California.

Araujo, R., Santoro, F. M., Brézillon, P., Borges, M. R., & Rosa, M. G. P. (2004). *Context Models for Managing Collaborative Software Development Knowledge.* Paper presented at the First International Workshop on Modeling and Retrieval of Context (KI 2004), MRC2004, Ulm, Germany.

Argyris, C., & Schön, D. A. (1978). *Organizational Learning: A Theory of Action Perspective*: Addison-Wesley.

Armitage, P., & Berry, G. (1994). *Statistical Methods in Medical Research* (3rd ed.): Blackwell Science Ltd.

Atkinson, C., Bayer, J., Bunse, C., Kamsties, E., Laitenberger, O., Laqua, R., et al. (2002). *Component-based Product Line Engineering with UML*. New York: Addison-Wesley.

Aviation Industry CBT Committee. (1998). AICC AGR010 - Web-based Computer Managed Instruction (CMI)Ver. 1.0, , from http://www.aicc.org/pages/down-docs-index.htm

Backhaus, K., Erichson, B., Plinke, W., & Weiber, R. (2005). *Multivariate Analysemethoden - Eine anwendungsorientierte Einführung* (11th ed.). Berlin: Springer.

Bannan-Ritland, B., Dabbagh, N., Murphy, K., & Wiley, D. A. (2002). Learning Object Systems as Constructivist Learning Environments: Related Assumptions, Theories, and Applications. *The Instructional Use of Learning Objects (AECT)*, 61-98.

Barker, N., News, R. J., Huitson, A., & Poloniecki, J. (1995). The two-period crossover trial. *BIAS, 9*, 67-116.

Baron, J. (2000). *Thinking and Deciding*. Cambridge, UK: Cambridge University Press.

Basili, V. R., Caldiera, G., & Cantone, G. (1992). A reference architecture for the component factory. *ACM Transactions on Software Engineering and Methodology, 1*(1), 53-80.

Basili, V. R., Caldiera, G., McGarry, F., Pajerski, R., Page, G., & Waligora, S. (1992). *The Software Engineering Laboratory - an Operational Software Experience Factory.* Paper presented at the International Conference on Software Engineering (ICSE).

Basili, V. R., Caldiera, G., & Rombach, H. D. (2002). Experience Factory. In J. J. Marciniak (Ed.), *Encyclopedia of Software Engineering* (2nd ed., Vol. 1, pp. 511-518). New York: John Wiley & Sons, Inc.

Basili, V. R., Caldiera, G., Rombach, H. D., & Van Solingen, R. (2002). The Goal Question Metric Approach. In J. J. Marciniak (Ed.), *Encyclopedia of Software Engineering* (2nd ed., Vol. 1, pp. 578-583). New York: John Wiley & Sons, Inc.

Basili, V. R., Daskalantonakis, M. K., & Yacobellis, R. H. (1994). Technology transfer at Motorola. *IEEE Software, 11*(2), 70-76.

Basili, V. R., & Rombach, H. D. (1991). Support for Comprehensive Reuse. *Journal of Software Engineering, 6*(5), 303 - 316.

Basili, V. R., & Weiss, D. (1984). A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering*(November), 728-738.

Baumgartner, P. (2000). Handeln und Wissen bei Schütz. In G. H. Neuweg (Ed.), *Wissen. Können. Reflexion* (pp. 9-26). Innsbruck, Wien: Studienverlag.

Bayer, J., Flege, O., & Gacek, C. (2000). *Creating Product Line Architectures*. Paper presented at the Int. Workshop on Software Architectures for Product Families (IW-SAPF-3).

Becker, M. (2004). *Anpassungsunterstützung in Softwareproduktfamilien.* Dissertation, University of Kaiserslautern, Kaiserslautern.

Biggerstaff, T. J. (1989). Design Recovery for Maintenance and Recovery. *IEEE Computer, 22*(7), 36-49.

Biggerstaff, T. J. (1991). *Panel: Software Reuse: Is it Delivering?* Paper presented at the ICSE.

Birk, A. (2001). A knowledge management infrastructure for systematic improvement in software engineering: Fraunhofer-IRB-Verl.

Birk, A., & Kröschel, F. (1999). *A Knowledge Management Lifecycle for Experience Packages on Software Engineering Technologies.* Paper presented at the Workshop on Learning Software Organizations.

Birk, A., & Tautz, C. (1998). *Knowledge Management of Software Engineering Lessons Learned.* Paper presented at the 10th International Conference on Software Engineering and Knowledge Engineering (SEKE'98).

Bloom, B. S. e., Engelhart, M. D., Furst, E. J., Hill, W. H., & Krathwohl, D. R. (1956). *Taxonomy of Educational Objectives; the Classification of Educational Goals* (1st ed.). New York: Longmans Green.

Bonar, J. G. (1988). *The Bite-Sized Architecture*. Pittsburgh, Pennsylvania: Learning Research and Development Center, University of Pittsburgh.

Bortz, J. (2005). *Statistik. Für Human- und Sozialwissenschaftler* (6th ed.). Heidelberg, New York: Springer.

Bortz, J., & Döring, N. (2001). *Forschungsmethoden und Evaluation. Für Human- und Sozialwissenschaftler* (4th ed.). Heidelberg, New York: Springer.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (1999). *How People Learn: brain, mind, experience, and school*. Washington DC: National Academy Press.

Bransford, J. D., Sherwood, R. D., Hasselbring, T. S., Kinzer, C. K., & Williams, S. M. (1990). Anchored Instruction: Why We Need it and How Technology Can Help. *Cognition, Education, Multimedia: Exploring Ideas in High Technology* 115-141.

Brézillon, P. (1999). Context in Problem Solving: A Survey. *The Knowledge Engineering Review, 14*(1), 1-34.

Brickley, D., & Guha, R. V. (2004). RDF Vocabulary Description Language 1.0: RDF Schema, from http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

Brown, J. S., Collins, A., & Duguid, P. (1989). *Situated Cognition and the Culture of Learning* (No. 481). Champaign, Ill.: University of Illinois at Urbana-Champaign.

Brown, M. (2004). Learning Spaces. In D. Oblinger, and Oblinger, J. (Ed.), *Educating the Net Generation*: Educause eBook.

Brusilovsky, P. (1996). Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction, 6*(2-3), 87-129.

Brusilovsky, P. (1998). *Adaptive educational systems on the world-wideweb: A review of available technologies.* Paper presented at the Workshop "WWW-Based Tutoring" at 4th International Conference on Intelligent Tutoring Systems, San Antonio, TX.

Brusilovsky, P. (2001). Adaptive Hypermedia. *User Modeling and User-Adapted Interaction, 11*, 87-110.

Brusilovsky, P., & Nijhawan, H. (2002). *A Framework for Adaptive E-Learning Based on Distributed Reusable Learning Activities*. Paper presented at the World Conference on E-Learning, E-Learn2002.

Brusilovsky, P., Pesin, L., & Zyryanov, M. (1993). Towards an adaptive hypermedia component for an intelligent learning environment. In L. J. Bass, J. Gornostaev & C. Unger (Eds.), *Lecture Notes in Computer Science, Human-Computer Interaction* (Vol. 753, pp. 348-358). Berlin: Springer-Verlag.

Brusilovsky, P., Schwarz, E., & Eklund, J. (1998). Web-based Education for All: A Tool for Developing Adaptive Courseware. *Computer Networks and ISDN Systems (proc. of seventh International World Wide Web Conference), 30*(1-7), 291-300.

Brusilovsky, P., & Vassileva, J. (2003). Course sequencing techniques for large-scale web-based education. *Int. J. Continuing Engineering Education and Lifelong Learning, 13*, 75-94.

Bühner, M. (2006). *Einführung in die Test- und Fragebogenkonstruktion* (2nd ed.). München, Germany: Pearson Studium.

Buser, K. P. (1995). *Dangers in Using ANCOVA to Evaluate Special Education Program Effects*. Paper presented at the Annual Meeting of the American Educational Research Association.

Card, D., & Comer, E. (1994). Why Do So Many Reuse Programs Fail? *IEEE Software*(September 1994), 114-115.

Center for Software Engineering. (1997). *COCOMO II Model Definition Manual*. Los Angeles, CA: Computer Science Department, University of Southern California.

Chang, S. G., Ahn, J. H. (2005). Product and process knowledge in the performance-oriented knowledge management approach. *Journal of Knowledge Management, Nr. 4*, 114-132.

Chen, W., & Mizoguchi, R. (2004). Learner Model Ontology and Learner Model Agent. In P. Kommers (Ed.), *Cognitive Support for Learning - Imagining the Unknown* (pp. 189-200): IOS Press.

Chi, M. T. H., Feltovich, P. J., & Glaser, R. (1991). Categorization and Representation Physics Problems by Experts and N0vices. *Cognitive Science, 5*, 121-152.

Cisco Systems, I. (1999, June 25). Cisco Systems Reusable Information Object Strategy Retrieved October 1, 2007, from http://www.cisco.com/warp/public/779/ibs/solutions/learning/whitepapers/el_cisco_rio.pdf

Clements, P., & Northrop, L. (2001). Software Product Lines: Practices and Patterns, *Addison Wesley* (3Rev Ed edition ed.).

Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences* (2nd ed ed.). Hillsdale, NJ: Lawrence Erlbaum Associates.

Collins, A., Brown, J. S., & Newman, S. E. (1989). Cognitive Apprenticeship: teaching the crafts of reading, writing and mathematics. In L. B. Resnick (Ed.), *Knowing, Learning and Instruction: Essays in honor of Robert Glaser* (pp. 453-494): Hillsdale, NJ: Lawrence Erlbaum Associates.

Conklin, J. (1987). Hypertext: An Introduction and Survey. *IEEE Computer*(20), 9.

Conlan, Lewis, D., Higel, S., O'Sullivan, D., & Wade, V. (2003). *Applying Adaptive Hypermedia Techniques to Semantic Web Service Composition*. Paper presented at the International Workshop on Adaptive Hypermedia and Adaptive Web-based Systems (AH 2003).

Conlan, O., Dagger, D., & Wade, V. (2002). *Towards a Standards-based Approach to e-Learning Personalization using Reusable Learning Objects*. Paper presented at the E-Learn 2002, World Conference on E-Learning in Corporate, Government, Healthcare and Higher Education.

Conradi, R. (1999). *From Software Experience Databases to Learning Organizations*. Paper presented at the SEKE 1999.

Consortium, R. P. (2005). RISE Homepage, from http://www.rise-it.info

Cook, T. D., & Campbell, D. T. (1979). *Quasi-Experimentation: Design and Analysis Issues for Field Settings*: Houghton Mifflin Company.

Coplien, J. O. (1998). *Multi-Paradigm Design for C++*: Addision Wesley.

Crowder, M. J., & Hand, D. J. (1990). *Analysis of Repeated Measures*. New York: Chapman & Hall.

Curtis, B. (1989). Cognitive Issues in Reusing Software Artifacts. In T. J. Biggerstaff & A. J. Perlis (Eds.), *Software Reusability. Vol.II: Applications and Experience* (pp. 269-287): Reading/Mass.

Curtis, B., Krasner, H., & Iscoe, N. (1988). A Field Study of the Software Design Process for Large Systems. *Communications of the ACM, 31*(11), 1268-1287.

Czarnecki, K., & Eisenecker, U. (2000). *Generative Programming - Methods, Tools, and Applications*. Boston, MA: Addison Wesley.

D'Alessandro, M., Iachini, P. L., & Martelli, A. (1993). *The Generic Reusable Component: An Approach to Reuse Hierarchical 00 Designs.* Paper presented at the Second International Workshop on Software Reusability, Lucca, Italy.

Dagger, D., Wade, V., & Conlan, O. (2003). *Towards "anytime, anywhere" Learning: The Role and Realization of Dynamic Terminal Personalization in Adaptive eLearning.* Paper presented at the Ed-Media 2003, Honolulu, Hawaii, USA.

Dalbey, J. (2003). Pseudocode Standard Retrieved June 1, 2008, from http://users.csc.calpoly.edu/~jdalbey/SWE/pdl_std.html

Daly, A., Brooks, A., Miller, J., Roper, M., & Wood, M. (1995). *The Effect of Inheritance on the Maintanability of Object-oriented Software: an Empirical Study.* Paper presented at the Int. Conf. on Software Maintenance.

Damiani, E., Fugini, G., & Bellettini, C. (1999). A hierarchy-aware approach to faceted classification of objected-oriented components. *ACM Trans. Softw. Eng. Methodol., 8*(3), 215-262.

Davenport, T. H., De Long, D. W., & Beers, M. C. (1998). Successful Knowledge Management Projects. *Sloan Management Review, Winter 1998, 39*(2), 43-57.

Davis, F. D. (1986). *A Technology Acceptance Model for Empirically Testing New End-User Information Systems: Theory and Results (PhD Thesis).* Massachusetts Institute of Technology.

Davis, F. D. (1989). Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology. *MIS Quarterly, 13*(3), 319-339.

De Bra, P., Aerts, A., Smits , D., & Stash, N. (2002). *AHA! Version 2.0: More Adaptation Flexibility for Authors*. Paper presented at the AACE ELearn2002 Conference.

De Bra, P., Houben, G. J., & Wu, H. (1999). *AHAM: A Dexter-Based Reference Model for Adaptive Hypermedia*. Paper presented at the ACM Conference on Hypertext and Hypermedia.

De Bra, P., Stash, N., & De Lange, B. (2003, May). *AHA! Adding Adaptive Behavior to Websites.* Paper presented at the NLUUG Conference, Ede, The Netherlands.

De Jong, T., & Ferguson-Hessler, M. G. M. (1996). Types and Qualities of Knowledge. *Educational Psychologist, 31*(2), 105-113.

Decker, B., Ras, E., Rech, J., Jaubert, P., & Rieth, M. (2007). Wiki-Based Stakeholder Participation in Requirements Engineering. *IEEE Software, 24*(2), 28-35.

Decker, B., Rech, J., Ras, E., Klein, B., & Hoecht, C. (2006). *Using Wikis to Manage Use Cases: Experience and Outlook.* Paper presented at the International Workshop on Learning Software Organizations and Requirements Engineering (LSO+RE 2006), Hannover, Germany.

Dehnbostel, P. (2001). Perspektiven für das Lernen in der Arbeit. In A. B. W. e.V. (Ed.), *Kompetenzentwicklung 2001. Tätigsein - Lernen - Innovation* (pp. 53-93). Münster.

Denaux, R., Dimitrova, D., & Aroyo, L. (2004). *Interactive Ontology-Based User Modeling for Personalized Learning Content Management*. Paper presented at the SWEL'04 Workshop (AH2004).

Devanbu, P., Brachmann, R. J., Selfridge, P. G., & Ballard, B. W. (1991). LaSSIE: A Knowledge-Based Software Information System. *Communications of the ACM, 34*, 35-49.

Dewe, B., Wiesner, G., Wittpoth, J. (2001). Professionswissen und erwachsenenpädagogisches Handeln Retrieved October 1, 2007, from http://www.die-bonn.de/esprid/dokumente/doc-2002/dewe02_01.pdf

Dewey, J. (1933). *How we think, a restatement of the relation of reflective thinking to the educative process*. Boston: New York [etc.] D.C. Heath and company.

Dewey, J. (1938). *Experience and Education*. New York: Collier Books.

Dey, A. (2000). *Providing Architectural Support for Building Context-Aware Applications.* PhD Thesis, College of Computing,Georgia Tech.

Dey, A. K. (2001). Understanding and Using Context. *Personal and Ubiquitous Computing, 5*(1), 4-7.

Dhungana, D. (2006). *Integrated Variability Modeling of Features and Architecture in Software Product Line Engineering*. Paper presented at the 21st International Conference on Automated Software Engineering (ASE 2006).

Dhungana, D., Kepler, J., Rabiser, R., & Grunbacher, P. (2007). *Decision-Oriented Modeling of Product Line Architectures*. Paper presented at the Sixth Working IEEE/IFIP Conference on Software Architecture (WICSA'07).

Díaz-Uriarte, R. (2002). Incorrect Analysis of Crossover Trials in Animal Behaviour Research. *Animal Behavior, 63*(4), 815-822.

Doan, B. L., Bourda, Y., & Dumitrascu, V. (2006, July 5-7). *A Semi-Automatic Tool using Ontology to Extract Learning Objects.* Paper presented at the Sixth International Conference on Advanced Learning Technologies, Kerkrade, The Netherlands.

Dochy, F., & Alexander, P. A. (1995). Mapping prior knowledge: A framework for discussion among researchers. *European Journal of Psychology of Education, 10*(3), 225-242.

Dodgson, M. (1993). Organizational Learning: A Review of Some Literatures. *Organization Studies, 14*(3), 375-394.

Dolog, P., Gavriloaie, R., Nejdl, W., & Brase, J. (2003). *Integrating Adaptive Hypermedia Techniques and Open RDF-Based Environments*. Paper presented at the Twelfth International World Wide Web Conference (WWW2003).

Dolog, P., Henze, N., Nejdl, W., & Sintek, M. (2003). *Towards the Adaptive Semantic Web*. Paper presented at the Principles and Practice of Semantic Web Reasoning (PPSWR103), .

Dusink, L., & Van Katwijk, J. (1995). *Reuse Dimensions.* Paper presented at the Symposium on Software Reusability, Seattle, Washington, USA.

Efimova, L., & Swaak, J. (2002). *KM and (e)-learning: towards an integral approach?* Paper presented at the EKMF.

Efimova, L., & Swaak, J. (2003). Converging Knowledge Management, Training and e-learning: Scenarios to make it work. *Journal of Universal Computer Science, 9*(6), 571-578.

Encyclopedia Britannica Online. Retrieved November 29, 2008, from http://www.britannica.com/

Endres, A., & Rombach, H. D. (2003). *A Handbook of Software and Systems Engineering : Empirical Observations, Laws, and Theories*. Harlow, England: Addison Wesley.

Enns, C. Z. (1993). Integrating Separate and Connected Knowing: The Experiential Learning Model. *Teaching of Psychology, 20(1)*, 7-13.

ergo-online. (2008). Beurteilung der Software-Ergonomie nach ISONORM Retrieved 2008-02-10, from http://www.ergo-online.de

Ericsson, K. A., Krampe, R. T., & Tesch-Römer, C. (1993). The Role of Deliberate Practice in the Acquisition of Expert Performance. *Psychological review. 100, no. 3*, 363-406.

Falbo, R., Borges, L., Valente F. (2004, September 8-9). *Using Knowledge Management to Improve Software Process Performance in a CMM Level 3 Organization.* Paper presented at the Fourth International Conference on Quality Software, Braunschweig.

Farmer, M., & Taylor, B. (2002). *A Creative Learning Environment (CLE) for Anywhere Anytime Learning.* Paper presented at the MLearn 2002, Birmingham, UK.

Faul, F. (2006). G*Power. University Kiel, Germany.

Felder, M., & Silverman, L. K. (1988). Learning and Teaching Styles in Engineering Education. *Engr. Education, 78*(7), 674-681.

Felder, R. M., & Henriques, E. R. (1995). Learning and Teaching Styles In Foreign and Second Language Education. *Foreign Language Annals, 28*, 21-31.

Feldmann, R. L. (1999). *On Developing a Repository Structure Tailored for Reuse with Improvement*. Paper presented at the Learning Software Organizations (LSO 1999).

Feldmann, R. L., Frey, M., Habetz, M., & Mendonca, M. (2000, July 6-8). *Applying Roles in Reuse Repositories.* Paper presented at the International Conference on Software Engineering and Knowledge Engineering (SEKE'2000), Chicago, USA.

Feldmann, R. L., Frey, M., & Mendonca, M. (2000, July 6-8). *Applying roles in reuse repositories.* Paper presented at the International Conference on Software Engineering and Knowledge Engineering (SEKE'2000), Chicago, USA.

Fiol, C. M., & Lyles, M. A. (1985). Organizational Learning. *Academy of Management Review, 10*(4), 803-813.

Fischer, G. (1987). Cognitive View of Reuse and Design. *IEEE Software*(July 1987), 60-72.

Fischer, G., Henninger, S., & Redmiles, D. (1991). *Cognitive Tools for Locating and Comprehending Software Objects for Reuse.* Paper presented at the 13th International Conference on Software engineering, Austin, Texas, USA.

Fisseni, H. (1997). *Lehrbuch der psychologischen Diagnostik* (2. Aufl. ed.). Göttingen: Hogrefe.

Flavell, J. H. (1992). Metakognition and cognitive monitoring: A new Area of cognitive developmental inquiry. *American Psychologist, 34*, 906-911.

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

Frakes, W. B., & Fox, C. J. (1995). Sixteen Questions About Software Reuse. *Communications of the ACM, 38*(6), 75-87.

Frakes, W. B., & Kang, K. (2005). Software Reuse Research: Status and Future. *IEEE Transactions on Software Engineering, 31*(7), 529-536.

Frakes, W. B., & Pole, T. P. (1994). An Empirical Study of Representation Methods for Reusable Software Components. *IEEE Transactions on Software Engineering, 20*(8), 617-630.

Frankola, K. (2001). Why Online Learners Dropout. *Workforce, 10*, 53-63.

Frasincar, F., & Houben, G. J. (2002). *Hypermedia Presentation Adaptation on the Semantic Web*. Paper presented at the Proceedings of the 2nd International Conference on Adaptive Hjpermedia and Aduptive Web-Based Systems (AH 2002).

Fraunhofer IFF, F. I. (2006). Wissensmanagement in produzierenden KMU Retrieved October 1, 2007, from http://www.wissensmanagement.fraunhofer.de/kurzversion_prowis_studie.pdf

Freeman, P. R. (1989). The Performance of the Two-Stage Analysis of Two-Treatment, Two-Period Crossover Trials. *Statistics in Medicine, 8*(12), 1421-1432.

Friesen, N. (2004). Three Objections to Learning Objects and E-learning Standards. In R. McGreal (Ed.), *Online Education Using Learning Objects* (pp. 59-70). London: Routledge.

Fröming, J., Korf, R., Fürstenau, D. (2005). Arbeitsbericht Knowledge Modelling and Description Language Retrieved October 1, 2007, from http://wi.uni-potsdam.de/hp.nsf/0/83C8D2924031BAE2C12570A10054EE58/$FILE/WI-2005-23.pdf

Gagné, R. M. (2005). *Principles of instructional design* (5th ed.). Belmont, CA: Thomson/Wadsworth.

Gamma, E. (1995). *Design patterns : elements of reusable object-oriented software*. Reading, Mass.: Addison-Wesley.

Gardner, H. (1983). *Frames of Mind: the theory of multiple intelligences*: Basic Books.

Gardner, H. E. (1999). Multiple Approaches to Understanding. In C. M. Reigeluth (Ed.), *Instructionl-Design Theories and Models: A New Paradigm of Instructional Theory* (Vol. II). Mahwah, NJ, USA: Lawarence Erlbaum Associates, Inc.

Gilbert, J. E., & Han, C. Y. (1999). *Arthur: Adapting Instruction to Accommodate Learning Style*. Paper presented at the World Conference of the WWW and Internet.

Goeken, M. (2005, September). Wissensarten und Techniken im Anforderungsmanagement. Retrieved October 1, 2007, from http://www.frankfurt-school.de/dms/publications_it_governance/Goeken_Informatik05/Goeken_Informatik05.pdf

Goodyear, P. (2005). Educational Design and Networked Learning: Patterns, Pattern Languages and Design Practice. *Australasian Journal of Educational Technology, 21*(1), 82-101.

Greaves, M. (2007). Semantic Web 2.0. *IEEE Intelligent Systems, 22*(2), 94-96.

Greenwald, A. (1976). Within-Subjects Designs: To Use of not to Use? *Psychological Bulletin, 83*(2), 314-320.

Grieve, A. P. (1985). A Bayesian Analysis of the Two-Period Crossover Design for Clinical Trials. *Biometrics, 41*, 979-990.

Grizzle, J. E. (1965). The Two-Period Change-over Design and its Use in Clinical Trials. *Biometrics, 21*, 314-320.

Gronau, N. (2006). Softwaregestütztes Staffing zur kompetenzorientierten Zusammenstellung von Projektteams Retrieved October 1, 2007, from http://www.competence-site.de/wissensmanagement.nsf/2DA1012F7887B360C125720C00305C4A/$File/softwaregest%C3%BCtztes%20staffing.pdf

Group, P. C. (2004). Wissen - ein Potenzial für Unternehmen? Retrieved October 1, 2007, from http://www.competence-site.de/wissensmanagement.nsf/103F3939DAB8E4BEC1256EF90030D7D6/$File/knowledge%20mgmt%20final%20report.pdf

Guide to the Software Engineering Body of Knowledge, SWEBOK. (2004). IEEE CS Press.

Guretzky, B. (2002). Wissensmanagement und Software Engineering - Expert Assistance Retrieved October 1, 2007, from http://www.community-of-knowledge.de/cp_artikel.htm?artikel_id=51

Haas, P. (2006). Wissensarten Retrieved October 1, 2007, from http://e-health-com.de/service/glossar/w/index_09107.html

Halasz, F., & Schwartz, M. (1994). The Dexter Hypertext Reference Model. *Communications of the ACM, 37*, 30-39.

Haley, T. J. (1996). Software process improvement at Raytheon. *IEEE Software, 13*(6), 33-41.

Hamilton, A. G. (1978). *Logic for Mathematicians*. Cambridge, UK: Cambridge University Press.

Hardman, L., Bulterman, C., & van Rossum, G. (1994). The Amsterdam Hypermedia Model. *Communications of the ACM, 37*(2), 50-62.

Henze, N. (2005). *Personal Readers: Personalized Learning Object Readers for the Semantic Web*. Paper presented at the 12th International Conference on Artificial Intelligence in Education (AIED2005).

Henze, N., & Nejdl, W. (2000). *Extendible Adaptive Hypermedia Courseware: Integrating Different Courses and Web Material*. Paper presented at the Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2000).

Henze, N., & Nejdl, W. (2001). Adaptation in Open Corpus Hypermedia. *International Journal of Artificial Intelligence in Education. Special Issue on Adaptive and Intelligent Web-Based Systems, 12*.

Henze, N., & Nejdl, W. (2004). A Logical Characterization of Adaptive Educational Hypermedia. *New Review of Hypermedia and Multimedia, 10*(1), 77-113.

Hills, M., & Armitage, P. (1979). The Two-Period Cross-Over Clinical Trial. *British Journal of Clinical Pharmacology, 8*, 7-20.

Hockemeyer, C., Held, T., & Albert, D. (1998). *RATH A Relational Adaptive Tutoring Hypertext WWW-Environment Based on Knowledge Space Theory*. Paper presented at the 4th International conference on Computer Aided Learning and Instruction in Science and Engineering.

Houdek, F., & Kempter, H. (1997). *Quality Patterns - An Approach to Packaging Software Engineering Experience.* Paper presented at the Proceedings of the 1997 Symposium on Software Reusability, Boston, Massachusetts, USA.

Houdek, F., Schneider, K., & Wieser, E. (1998). *Establishing experience factories at Daimler-Benz an experience report.* Paper presented at the Software Engineering, 1998. Proceedings of the 1998 (20th) International Conference on.

Houle, C. (1980). *Continuing Learning in the Professions*. San Francisco: Jossey-Bass.

Humphrey, W. S. (1991). Software and the factory paradigm. *Software Engineering Journal, 6*(5), 370-376.

IEEE Learning Technology Standard Comittee. (2002). 1484.12.1, IEEE Standard for Learning Object Metadata Retrieved 6 March, 2008, from http://ltsc.ieee.org/wg12/

Ilin, D. (2008). *Generierung von adaptiven Lernräumen für erfahrungsbasiertes Lernen.* Diploma Thesis, Fachbereich Informatik und Mikrosystemtechnik, Fachhochschule Kaiserslautern, Kaiserslautern.

IMS Global Learning Consortium, I. (2001, June). Learning Resource Meta-data Specification Retrieved October, 2007, from http://www.imsglobal.org/metadata/index.html

International Organization of Standardization. (2008). DIN EN ISO 9241-110 Grundsätze der Dialoggestaltung Retrieved 2008-02-10, from http://www.iso.org

Jedlitschka, A., Althoff, K.-D., Decker, B., Hartkopf, S., & Nick, M. (2001). *Corporate Information Network (COIN): The Fraunhofer IESE Experience Factory.* Paper presented at the International Conference on Case-Based Reasoning (ICCBR), Vancouver, Canada.

Jedlitschka, A., Althoff, K.-D., Decker, B., Hartkopf, S., Nick, M., & Rech, J. (2002). The Fraunhofer IESE Experience Management System. *Künstliche Intelligenz, 16*(1), 70-73.

Johannson, C., Hall, P., & Coquard, M. (1999). "*Talk to Paul and Peter - They are Experienced*" - *The Experience Engine in a Nutshell.* Paper presented at the Learning Software Organisation (1999).

Joint Task Force on Computing Curricula. (2004). *Software Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*.

Jonassen, D. (1999). Designing Constructivist Learning Environments. In C. M. Reigeluth (Ed.), *Instructional Design Theories and Models: A New Paradigm of Instructional Theory* (Vol. II, pp. 215-240). Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc.

Jones, B., & Kenward, M. G. (2003). *Design and Analysis of Cross-Over Trials* (2nd ed.). Boca Raton, FL: Chapman and Hall/CRC Press.

Jones, B., & Lewis, A. (1995). The Case for Cross-over Trials in Phase III. *Statistics in Medicine, 14*, 1025-1038.

Joos, R. (1994). Software Reuse at Motorola. *IEEE Software*(September 1994), 42-47.
Jungmann, M., & Paradies, T. (1997). *Adaptive Hypertext in Complex Knowledge Domains*. Paper presented at the Flexible Hypertext Workshop (Hypertext´97).
Juristo, N., & Moreno, A. M. (2001). *Basics of Software Engineering Experimentation* (1st ed.). Berlin: Springer.
Kamel, A., M., C., & Sorenson, P. (2001, 30 July - 2 August 2001). *Building an Experience-Base for Product-line Software Development Process.* Paper presented at the Fourth International Conference on Case-Based Reasoning, Vancouver, British Columbia, Canada.
Kang, K., Cohen, S., Hess, J., Novak, W., & Peterson, A. (1990). *Feature-Oriented Domain Analysis (FODA) Feasibility Study*. Pittsburgh: Software Engineering Institute.
Karampiperis, P., & Sampson, D. (2005). Adaptive Learning Resources Sequencing in Educational Hypermedia Systems. *Journal of Educational Technology & Society, Nr. 8*, 128-147.
Kärger, P. (2006). *Ontologie-basierter Mediator zum Zugriff auf heterogene und verteilte Lerninhalte.* diploma thesis, Universität des Saarlandes.
Karlsson, E. A. (1995). *Software Reuse: A Holistic Approach*: Wiley and Sons.
Kasunic, M. (1992). *A Reuse-based Software Development Methodology. Process Guide 1.0*: Software Productivity Consortium Services Corporation.
Kim, Y.-G., Kim, J.-W., Shin, S.-O., & Baik, D.-K. (2006). *Managing Variability for Software Product-Line*. Paper presented at the Fourth International Conference on Software Engineering Research, Management and Applications (SERA 2006).
Kirk, R. E. (1995). *Experimental Design: Procedures for the Behavioral Science* (3rd ed.). Pacific Grove: Brooks/Cole Publishing Company.
Kitchenham, B. A., Fry, J., & Linkman, S. (2003). *The Case Against Cross-over Designs in Software Engineering.* Paper presented at the 11th annual international workshop on Software Technology and Engineering Practice (STEP2004).
Knolmayer, G. F. (2003). *Decision support models for composing and navigating through e-learning objects*. Paper presented at the 36th IEEE Annual Hawaii International Conference on System Sciences.
Koch, N. P. (2000). *Software Engineering for Adaptive Hypermedia Systems. Reference Model, Modeling Techniques and Development Process.* PhD Thesis, Ludwig-Maximilians-Universität, Munich, Germany.
Kolb, D. A. (1984). *Experiential learning: experience as the source of learning and development*. Englewood Cliffs, N.J.: Prentice-Hall.
Kolb, D. A., & Fry, R. (1975). Toward an applied theory of experiential learning. In C. Cooper (Ed.), *Theories of Group Process*. London: John Wiley.
Koper, R. (2003). Combining reusable learning resources and services to pedagogical purposeful units of learning. In *Reusing Online Resources: A Sustainable Approach to eLearning* (pp. 46-59): Kogan Page.
Laitenberger, O. (2000). *Cost-effective Detection of Software Defects Through Perspective-based Inspections.* PhD Thesis, University of Kaiserslautern, Kaiserslautern.
Langenbacher, G. (2002). *Verlebensweltlichung der Systeme durch Wissensmanagement.* doctoral thesis, Fernuniversität Hagen.
Lawton, G. (2001). Knowledge Management: Ready for Prime Time? *Computer, 34*(2), 12-14.
Lee, N.-Y., & Litecky, C. R. (1997). An Empirical Study of Software Reuse with Special Attention to Ada. *IEEE Transactions on Software Engineering, 23*(9), 537-549.
Lenat, D. (1998). The dimensions of context-space [Electronic Version] from http://www.cyc.com/context-space.doc.
Lenk, K., Wengelowski, P. (2002). Wissensmanagement für das Verwaltungshandeln Retrieved October 1, 2007, from http://www.uni-oldenburg.de/fk2/InstRW/vw/download/wm.pdf
Leuf, B., & Cunningham, W. (2001). *The Wiki Way. Collaboration and Sharing on the Internet*: Addison-Wesley.

Lewin, K. (1951). *Field Theory in Social Science*. New York: Harpers & Brothers.
Lieberman, H., & Selker, T. (2000). Out of context: Computer systems that adapt to, and learn from, context. *IBM Systems Journal, 39*(34), 617-632.
Likert, R. (1932). A Technique for the Measurement of Attitude. *Archive of Psychology, 22*(140).
Lindvall, M., Frey, M., Costa, P., & Tesoriero, R. (2001, 12-13 Sept. 2001). *Lessons learned about structuring and describing experience for three experience bases.* Paper presented at the Third International Workshop on Advances in Learning Software Organizations (LSO 2001), Kaiserslautern, Germany.
Loftin, L. B., & Madison, S. Q. (1991). The Extreme Dangers of Covariance Corrections. In B. Thompson (Ed.), *Advances in Education Research: Substantive Findings, Methodological Developments* (Vol. 1, pp. 133-147). Greenwich, CT: JAI Press.
Macdonald, F., & Miller, J. (1998). *A Comparision of Tool-based and Paper-based Software Inspections*: Department of Computer Science, University ogf Strathclyde, UK.
Maiden, N. A., & Sutcliffe, A. (1993, March 24-26). *People-oriented Software Reuse: the Very Thought.* Paper presented at the Second International Workshop on Software Reuse, Italy
Manola, F., & Miller, E. (2004). RDF Primer, from http://www.w3.org/TR/2004/REC-rdf-primer-20040210/
Martin, H. (1995). *CeA - Computergestützte erfahrungs-geleitete Arbeit*. Berlin/Heidelberg/New York: Springer.
Mason, R. M. (1993). *Strategic Information Systems: Use of Information Technology in a Learning Organization*. Paper presented at the 26th Hawaii International Conference on System Sciences.
Mayer, R. E. (1999). Designing Instruction for Constructivist Learning. In C. M. Reigeluth (Ed.), *Instructionl-Design Theories and Models: A New Paradigm of Instructional Theory* (Vol. II). Mahwah, NJ, USA: Lawarence Erlbaum Associates, Inc.
McIllroy, M. D. (1968). *Mass-produced Software Components.* Paper presented at the NATO Conference on Software Engineering, Garmisch, Germany.
Meder, N. (2001). Didaktische Ontologien Retrieved October 1, 2007, from http://cweb.uni-bielefeld.de/agbi/cgi-bin-noauth/cache/VAL_BLOB/167/167/63/did.pdf
Melis, E., Andrès, E., Büdenbender, J., Frishauf, A., Goguadse, G., Libbrecht, P., et al. (2001). ActiveMath: A Web-based Learning Environment. *International Journal of Artificial Intelligence in Education, 12*(4), 385-407.
Memmel, M., Ras, E., Jantke, J. K., & Yacci, M. (2006). Approaches to Learning Object Oriented Instructional Design. In A. Koohang & K. Harman (Eds.), *Principles and Practices of the Effective Use of Learning Objects* (pp. 281-326). Santa Rosa, California: Informing Science Press.
Memmel, M., Ras, E., Weibelzahl, S., & Burgos, D. l. (2006). *Joint International Workshop on Professional Learning, Competence Development and Knowledge Management - LOKMOL and L3NCD.* Paper presented at the EC-TEL2006, Crete, Greece.
Memmel, M., Ras, E., Wolpers, M., & Van Assche, F. (2007). *Proceedings of the 3rd Workshop on Learner-Oriented Knowledge Management & KM-Oriented E-Learning (LOKMOL 2007)*, Crete, Greece.
Merriënboer, J. J. G. v. (1997). *Training complex cognitive skills : a Four-Component Instructional Design model for technical training*. Englewood Cliffs, N.J.: Educational Technology Publications.
Merrill, M. D. (2000). *First principles of instruction.* Paper presented at the International conference of the Association for Educational Communications and Technology (AECT), Denver.
Mili, H., Mili, A., Yacoub, S., & Addy, E. (1995). *Reuse-Based Software Engineering - Techniques, Organization and Measurement*: John Wiley and Sons.
Mills, M., & Armitage, P. (1979). The two-period cross-over clinical trial. *British Journal of Clinical Pharmacology, 8*, 7-20.

Milosavljevic, M. (1997). *Augmenting the User's Knowledge via Comparison.* Paper presented at the 6th International Conference on User Modelling, Wien.
Mitchell, K. (2002). *Supporting the Development of Mobile Context-Aware Computing.* Ph. D. Thesis, University of Lancaster.
Mittelmann, A. (2005, January 9). Wissensmanagement. Grundlagen Retrieved October 1, 2007, from http://www.artm-friends.at/am/km/km-d/km-basics-d.html
Mödritscher, F. (2005). Praxisorientierte Aspekte des Wissensmanagements. *Journal of Applied Knowledge Management, Nr. 1*, 17-40.
Muthig, D. (2002). *A Lightweight Approach Facilitating an Evolutionary Transition Towards Software Product Lines.* PhD Thesis, University of Kaiserslautern, Kaiserslautern.
Natali, A., Falbo, R. (2002). Knowledge Management in Software Engineering Environments Retrieved October 1, 2007, from http://www.inf.ufes.br/~falbo/download/pub/Sbes2002.pdf
Naur, P., & Randell, B. (1968). *Software Engineering: Report of a conference.* Paper presented at the NATO Conference on Software Engineering, Garmisch, Germany.
Neuweg, G. H. (2000). Können und Wissen. In G. H. Neuweg (Ed.), *Wissen. Können. Reflexion* (pp. 65-82). Innsbruck, Wien: Studienverlag.
Nevis, E. C., DiBella, A. J., & Gould, J. M. (1995). Understanding Organisations as Learning Systems. *Sloan Management Review, 36*(2), 73-85.
Nickerson, R. S., Perkins, D. N., & Smith, E. E. (1985). *The Teaching of Thinking*: L. Erlbaum Associates.
Nonaka, I., & Takeuchi, H. (1995). *The Knowledge-Creating Company*. New York: Oxford University Press.
Norman, D. A. (1988). *The Psychology of Everday Things*. New York: Basic Books.
North, K. (2002). *Wissensorientierte Unternehmensführung : Wertschöpfung durch Wissen* (3., aktualisierte und erw. Aufl. ed.). Wiesbaden: Gabler.
O'Reilly, T. (2005). Web 2.0: Compact Definition? Retrieved May 29, 2008, from http://radar.oreilly.com/archives/2005/10/web-20-compact-definition.html
Oppermann, R. (1994). *Adaptive User Support Pod: Ergonomic Design of Manually and Automatically Adaptable Software*: Lawrence Erlbaum Associates.
Oppermann, R., Rashev, R., & Kinshuk, K. (1997). Adaptability and Adaptivity in Learning Systems. In A. Behrooz (Ed.), *Knowledge Transfer* (Vol. 2, pp. 173-179): London: Pace.
Orr, J. E. (1996). *Talking about Machines - an Ethnography of a Modern Job*: IRL Cornell Press.
Pascoe, J. (2001). *Context-Aware Software.* PhD Thesis, University of Kent Canterbury, U.K.
Pech, D. (2007). *Variability Management Support for Large-Scale Software Product Lines.* Diploma Thesis, University of Kaiserslautern, Kaiserslautern.
Pérez, T., Lopistéguy, P., Gutiérrez, J., & Usandizaga, I. (1995). *HyperTutor: From Hypermedia to Intelligent Adaptive Hypermedia*. Paper presented at the World Conference on Educational Multimedia and Hypermedia.
Piaget, J. (1971). *Psychology and Epistemology*. Middlesex, England: Penguin Books.
Piaget, J. (1976). *Die Äquilibration der kognitiven Strukturen*. Stuttgart: Klett.
Polanyi, M. (1966). *The Tacit Dimension*: Routledge and Kegan Paul.
Poloniecki, J., & Daniel, D. (1981). Further Analysis of the Hills and Armitage Enursesis data. *The Statistician, 20*, 276-285.
Probst, G., Raub, S., & Romhardt, K. (1998). *Wissen managen: wie Unternehmen ihre wertvollste Ressource optimal nutzen*. Frankfurt am Main: Gabler Verlag.
Rambow, R., & Bromme, R. (2000). Der „reflective practitioner" und die Kommunikation mit Laien. In G. H. Neuweg (Ed.), *Wissen. Können. Reflexion* (pp. 245-263). Innsbruck, Wien: Studienverlag.
Ras, E. (2009a). *Dissertation Eric Ras - Annex 1: Explorative Studies* (No. IESE-Report 001.09/E). Kaiserslautern: Fraunhofer IESE.

Ras, E. (2009b). *Dissertation Eric Ras - Annex 2: Materials of the Empirical Studies* (No. IESE-Report 002.09/E). Kaiserslautern: Fraunhofer IESE.

Ras, E., Carbon, R., Decker, B., & Rech, J. (2007). Experience Management Wikis for Reflective Practice in Software Capstone Projects. *IEEE Transactions on Education, 50*(4), 312-320.

Ras, E., Memmel, M., Lindstaedt, S., Ley, T., & Albert, D. (2008). *Special Track on Intelligent Assistance for Self-Directed and Organizational Learning (IWL-LOKMOL '08)*, Graz, Austria.

Ras, E., Memmel, M., & Weibelzahl, S. (2005). *Integration of E-Learning and Knowledge Management - Barriers, Solutions and Future Issues.* Paper presented at the Professional Knowledge Management (WM2005), Kaiserslautern, Germany.

Ras, E., Rech, J., & Decker, B. (2006, Sept. 6-8). *Workplace Learning in Software Engineering Reuse.* Paper presented at the I-KNOW '06, Special track on "Integrating Working and Learning", Graz, Austria.

Rausch, A., Broy, M., Bergner, K., Höhn, R., & Höppner, S. (2007). *Das V-Modell XT. Grundlagen, Methodik und Anwendungen*. Heidelberg: Springer.

Rech, J., & Ras, E. (2007). Systematische Aggregation von Erfahrungen im Erfahrungsmanagement. *Künstliche Intelligenz, 21*(4), 16-22.

Rech, J., Ras, E., & Decker, B. (2006a). *Intelligente Assistenz in der Sotwareentwicklung 2006: Zusammenfassung der Ergebnisse*. Kaiserslautern: Fraunhofer IESE.

Rech, J., Ras, E., & Decker, B. (2006b). *RISE: Schlussbericht des Fraunhofer IESE zum Projekt RISE* (Technical Report No. IESE-144.06/D). Kaiserslautern: Fraunhofer IESE.

Rech, J., Ras, E., & Decker, B. (2007a). Intelligent Assistance in German Software Development: A Survey. *IEEE Software, 24*(4), 72-79.

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

Reed, J. F. (2004). Analysis of Two-treatment, Two-period Crossover Trials in Emergency Medicine. *Annals of emergency medicine, 43*(1), 54-58.

Reeves, B. (1990). *Finding and Choosing the Right Object in a Large Hardware Store - an Empirical Study of Cooperative Problem Solving Among Humans*. Boulder, CO: Department of Computer Science, University of Colorado.

Reigeluth, C. M. (1999). The Elaboration Theory: Guidance for Scope and Sequence Decisions. In C. M. Reigeluth (Ed.), *Instructional-Design Theories and Models: A New Paradigm of Instructional Theory* (Vol. II). Mahwah, NJ, USA: Lawrence Erlbaum Associates Inc.

Reigeluth, C. M., & Moore, J. (1999). Cognitive Education and the Cognitive Domain. In C. M. Reigeluth (Ed.), *Instructional-Design Theories and Models: A New Paradigm of Instructional Theory* (Vol. II). Mahwah, NJ, USA: Lawrence Erlbaum Associates Inc.

Renkl, A. (1996). Träges Wissen. Wenn gelerntes nicht genutzt wird. *Psychologische Rundschau, 47*, 78-92.

Rombach, H. D. (1991). Software Reuse: A Key to the Maintenance Problem. *Information and Software Technology, 33*(1), 86-92.

Rombach, H. D. (2000). *Fraunhofer: the German Model for Applied Research and Technology Transfer.* Paper presented at the International Conference on Software Engineering (ICSE 2000).

Rombach, H. D., Birk, A., Broeckers, A. C., Lott, M., & Verlage, M. (1994). *Qualitaetsorientierte, prozess-sensitive Softwareentwicklungsumgebungen im MVP-Project (in German)*. Kaiserslautern, Germany: Computer Science Department, University of Kaiserslautern.

Rosa, M. G. P., Borges, M. R. S., & Santoro, F. M. (2003, Sept. 28 - Oct. 2). *A Conceptual Framework for Analyzing the Use of Context in Groupware.* Paper presented at the 9th International Workshop on Groupware: Design, Implementation, and Use, Autrans, France.

Rose, H. (1991). Bedeutung des Erfahrungswissens für die Bedienung von CNC-Maschinen. *Zeitschrift für wirtschaftliche Fertigung und Automatisierung, 1, 1986*, 45-48.

Rosnow, R. L., & Rosenthal, R. (1996). Computing Contrasts, Effect Sizes, and Counternulls on other People's Published Data: General Procedures for Research Consumers. *Psychological Methods, 1*, 331-340.

Rothenberger, M. A., Dooley, K. J., Kulkarni, U. R., & Nada, N. (2003). Strategies for Software Reuse: a Principal Component Analysis of Reuse Practices. *IEEE Transactions on Software Engineering, 29*(9), 825-837.

Ruhe, G., & Bomarius, F. (1999). *Learning software organizations (LSO): methodology and applications.* Paper presented at the 11th International Conference on Software Engineering and Knowledge Engineering, SEKE'99, Kaiserslautern, Germany.

Rus, I., & Lindvall, M. (2002). Knowledge Management - Knowledge Management in Software Engineering - Guest Editors' Introduction. *IEEE Software, 19*(3), 26-38.

Rus, I., Lindvall, M., Sinha, S. (2001, November 29). Knowledge Management in Software Engineering Retrieved October 1, 2007, from https://www.sdn.sap.com/irj/sdn/go/portal/prtroot/docs/library/uuid/6339b090-0201-0010-3099-d539710c0468

Ryle, G. (1984). *The Concept of Mind*: University of Chicago Press.

Schank, R. C., Berman, T. R., & Macpherson, K. A. (1999). Learning By Doing. In C. M. Reigeluth (Ed.), *Instructional-Design Theories and Models: a New Paradigm of Instructional Theory* (Vol. II). Mahwah, NJ, USA: Lawarence Erlbaum Associates, Inc.

Schilit, W. N. (1995). *A System for Context-Aware Mobile Computing.* c, Columbia University, New York.

Schmidt, A. (2002). *Ubiquitous Computing - Computing in Context.* PhD Thesis, Lancaster University.

Schmidt, A. (2005). *Bridging the Gap Between Knowledge Management and E-Learning with Context-aware Corporate Learning Solutions.* Paper presented at the First Workshop on Learner-Oriented Knowledge Management & KM-oriented E-Learning (LOKMOL 2005) at Professional Knowledge Management Conference (WM2005), Kaiserslautern, Germany.

Schneider, K. (2000). *LIDs: A Light-Weight Approach to Experience Elicitation and Reuse.* Paper presented at the Second International Conference on Product Focused Software Process Improvement, Oulu, Finland.

Schneider, K., Hunnius, J.-P., & Basili, V. (2002). Experience in Implementing a Learning Software Organization. *IEEE Software*(May-June 2002), 46-49.

Schön, D. A. (1990). *Educating the Reflective Practitioner: Toward a New Design for Teaching and Learning in the Professions* (1st ed.). San Francisco: Jossey-Bass.

Schön, D. A. (1995). *The reflective practitioner: how professionals think in action*. London: Arena.

Schraefel, M. C., Carr, L., & De Roure, D. (2004). You've Got Hypertext. *Digital Information, 5*(1), 253.

Schütz, A. (1981). *Theorie der Lebensformen*. Frankfurt am Main: Suhrkamp.

SCORM. (2006). SCORM 2004, 2nd Edition Overview page Retrieved 2006/04/10, from http://www.adlnet.org/scorm/index.cfm

Selby, R. (1988). Empirically Analyzing Software Reuse in a Production Environment. In W. Tracz (Ed.), *Software Reuse: Emerging Technology* (pp. 176-189): IEEE Computer Society Press.

Self, J. (1992). *Computational Mathetics: the missing link in Intelligent Tutoring Systems re-search? Directions in Intelligent Tutoring Systems* (No. 91).

Semantic MediaWiki. Retrieved November 29, 2008, from http://www.mediawiki.org/

Senge, P. M. (1990). *The Fifth Discipline: the Art and Practice of the Learning Organization* (1st ed.). New York: Doubleday/Currency.

Senn, S. (1993). *Cross-over studies in clinical research*. West Sussex, England: John Wiley & Sons Ltd.

Senn, S., & Hildebrand, H. (1991). Cross-over trials, degree of freedom, the carry over effect and its dual. *Statistics in Medicine, 10*(1361-74).

Shapiro, S. S., & Wilk, M. B. (1965). An Analysis of Variance Test for Normality (complete samples). *Biometrika, 52*(3 and 4), 591-611.

Sharples, M. (2000). The Design of Personal Mobile Technologies for Lifelong Learning. *Computers and*

Educational Psychologist, 34, 177-193.

Siemens, G. (2004). Connectivism: A Learning Theory for the Digital Age Retrieved 6 June 2008, from http://www.elearnspace.org/Articles/connectivism.htm

Simon, H. A. (1981). *The Sciences of the Artificial*. Cambridge, MA: MIT Press.

Simons, C. L., Parmee, I. C., & Coward, P. D. (2003). 35 years on: to what extent has software engineering design achieved its goals? *IEE Proceedings Software, 150*(6), 337-350.

Smith, M. K., Welty, C., & McGuinness, D. L. (2004). OWL Web Ontology Language Guide, from http://www.w3.org/TR/2004/REC-owl-guide-20040210/

Snedecor, G. W., & G., C. W. (1989). *Statistical Methods* (Eighth Edition ed.): Iowa State University Press.

Specht, M. (1998). *Adaptive Methoden in computerbasierten Lehr/Lernsystemen.* PhD Thesis, Universität Trier, Trier.

Specht, M., & Oppermann, R. (1998). ACE - Adaptive Courseware Environment. *The New Review of Hypermedia and Multimedia, 4*, 141-161.

Special Issue on Software Product Lines. (2002). *IEEE Software, 19*(4).

Steiger, C. (2000). *Wissensmanagement in Beratungsprojekten auf Basis innovativer Informations- und Kommunikationstechnologien: Das System K3.* doctoral thesis, Universität Paderborn.

Stenmark, D. (2001). *The Relationship between Information and Knowledge.* Paper presented at the IRIS-24, Ulvik, Norway.

Sveiby, K. E. (1997). *The New Organizational Wealth: Managing and Measuring Knowledge-Based Assets*: Berrett-Koehler Publishers.

Tautz, C. (2001). *Customizing Software Engineering Experience Management Systems to Organizational Needs.* PhD Thesis in Experimental Software Engineering, University of Kaiserslautern, Kaiserslautern.

Tennyson, R. D., & Rasch, M. (1988). Linking Cognitive learning theory to instructional prescriptions. *Instructional Science, 17*, 369-385.

TikiWiki. Retrieved November 29, 2008, from http://de.tikiwiki.org/

Tochtermann, K., & Dittrich, G. (1996). The Dortmund family of Hypermedia Models. *Journal of Universal Computer Science, 2*(1).

Trapp, M. (2002). *A Flexible Approach for Coupling Experience Base Requirements and Applicable Schema Building Blocks.* Diploma Thesis, University of Kaiserslautern, Kaiserslautern, Germany.

Trapp, M. (2005). *Modeling Adaptation Behavior of Adaptive Embedded Systems.* PhD Thesis, University of Kaiserslauten, Kaiserslautern.

Tsandilas, T., & Schraefel, M. C. (2004). Usable adaptive hypermedia systems. *New Review of Hypermedia and Multimedia, 10*(1), 5-29.

Van Gurp, J., Bosch, J., & Svahnberg, M. (2000). *Managing Variabilities in Software Product Lines*. Paper presented at the Landelijk Architectuur Congress 2000.

Van Gurp, J., Bosch, J., & Svahnberg, M. (2001). *On the Notion of Variability in Software Product Lines*. Paper presented at the Working IEEE / IFIP Conference on Software Architecture (WICSA 2001).

Venkatesh, V., Morris, M. G., Davis, G. B., & Davis, F. D. (2003). User Acceptance of Information Technology: Toward a Unified View. *MIS Quarterly, 27*(3), 425-478.

Verlage, M. (1996). *About Views for Modeling Software Processes in a Role-Specific Manner*. Paper presented at the Joint Proc. of the SIGSOFT ´96 Workshops.

Wasson, B. (1996). *Instructional Planning and Contemporary Theories of Learning: is this a Self-contradiction?* Paper presented at the European Conference on Artificial Intelligence in Education, Lisbon, Portugal.

Weber, G., Kuhl, H.-C., & Weibelzahl, S. (2001). *Developing Adaptive Internet Based Courses with the Authoring System Netcoach*. Paper presented at the Proceedings of the Third Workshop on Adaptive Hypermedia (AH2001)

Weber, G., & Specht, M. (1997). *User Modeling and Adaptive Navigation Support in WWW-based Tutoring Systems*. Paper presented at the Sixth International Conference on User Modeling (UM97).

Weber, R., Aha, D. W., & Becerra-Fernandez, I. (2001). Intelligent lessons learned systems. *Expert systems with applications. 20, no. 1*, 94-100.

Weber, S., Thomas, L., Armbrust, O., Ras, E., Rech, J., Uenalan, Ö., et al. (2008). *A Software Organization Platform (SOP)*. Paper presented at the 10th Workshop on Learning Software Organizations (LSO 2008).

Weller, M., Pegler, C., & Mason, R. (2003). Working with learning objects - some pedagogical suggestions, from http://iet.open.ac.uk/pp/m.j.weller/pub/

Winer, B. J., Brown, D. R., & Michels, K. M. (1991). *Statistical Principles in Experimental Design* (3rd ed.): McGraw Hill.

Wohlin, C., Runeson, P., M., H., Ohlsson, M. C., Regnell, B., & Wesslen, A. (1999). *Experimentation in Software Engineering: An Introduction*: Springer.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: theory and practice. *Knowledge Engineering Review, 10*(2), 115-152.

Wu, H., & De Bra, P. (2001, October 23-26). *Sufficient Conditions for Well-behaved Adaptive Hypermedia Systems.* Paper presented at the First Asia-Pacific Conference on Web Intelligence, Maebashi City, Japan.

Wu, H., Houben, G.-J., & De Bra, P. (1998). *AHAM: A Reference Model to Support Adaptive Hypermedia Authoring*. Paper presented at the Conference on Information Science (InfWet 1998).

Yacci, M. (1999). The Knowledge Warehouse: Reusing Knowledge Components. *Performance Improvement Quarterly, 12*(3), 132-140.

Ziadi, T., Jézéquel, J., & Fondement, F. (2003). *Product Line Derivation with UML*. Paper presented at the Software Variability Management Workshop.

Related Publications

1. Diploma thesis and technical reports

Ras, E. (2009). *Dissertation Eric Ras - Annex 1: Explorative Studies* (No. IESE-Report 001.09/E). Kaiserslautern: Fraunhofer IESE.

Ras, E. (2009). *Dissertation Eric Ras - Annex 2: Experimental Materials of a Controlled Experiment about the Effect of Learning Spaces on Software Engineering Experience Reuse* (No. IESE-Report 002.09/E). Kaiserslautern: Fraunhofer IESE.

Rech, J., Decker, B., & Ras, E. (2006). *Intelligente Assistenz in der Softwareentwicklung 2006: Zusammenfassung der Ergebnisse* (No. IESE Report: IESE-046.06/D). Kaiserslautern: Fraunhofer IESE.

Ras, E. (2000). *Reuse of Know-How for Software Engineering Training and Education.* Published Diploma Thesis, University of Kaiserslautern.

2. Conference and workshop publications

Ras, E. (2008). *Improving Reuse of Experience Packages Through Learning Spaces: A Controlled Experiment.* Paper presented at the 2nd International Symposium on Empirical Software Engineering and Measurement (ESEM 2008), Kaiserslautern, Germany.

Ras, E., & Ilin, D. (2008). *Using Decision Models for the Adaptive Generation of Learning Spaces.* Paper presented at the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems 2008 (AH 2008), Hanover, Germany.

Ras, E., Rech, J., & Weber, S. (2008). *Collaborative Authoring of Learning Elements for Adaptive Learning Spaces*. Paper presented at the Workshop on Authoring of Adaptive and Adaptable Hypermedia (AAAH2008) at AH2008, Hanover, Germany.

Ras, E., & Rech, J. (2008). *Improving Knowledge Acquisition in Capstone Projects Using Learning Spaces for Experiential Learning*. Paper presented at the 21st Conference on Software Engineering Education and Training (CSEE&T 2008), Charleston, SC, USA.

Weber, S., Thomas, L., & Ras, E. (2008). *Investigating the Suitability of Mashups for Informal Learning and Personal Knowledge Management*.

Paper presented at the *Workshop on Mash-Up Personal Learning Environments (MUPPLE08)*, Maastricht, The Netherlands.

Weber, S., Thomas, L., Armbrust, O., Ras, E., Rech, J., Uenalan, Ö., et al. (2008). *A Software Organization Platform (SOP)*. Paper presented at the 10th Workshop on Learning Software Organizations (LSO 2008), Rome Italy.

Ras, E. (2007). *Resolving Variations in Learning Spaces for Experiential Learning*. Paper presented at the Second European Conference on Technology Enhanced Learning (ECTEL 2007). from http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-280/, Maastricht, The Netherlands.

Ras, E., Rech, J., & Decker, B. (2006). *Workplace Learning in Software Engineering Reuse.* Paper presented at the I-KNOW '06, Special track on "Integrating Working and Learning", Graz, Austria.

Ras, E., Memmel, M., & Weibelzahl, S. (2005). Integration of E-Learning and Knowledge Management - Barriers, Solutions and Future Issues. Paper presented at the Professional Knowledge Management (WM2005), Kaiserslautern, Germany.

Ras, E. (2005). *Just-in-Time Learning With Situational Content Objects.* Paper presented at the E-Learn 2005, World Conference on E-Learning in Corporate, Government, Healthcare & Higher Education, Vancouver, Canada.

Decker, B., Ras, E., Rech, J., Klein, B., & Hoecht, C. (2005). *Self-organized Reuse of Software Engineering Knowledge Supported by Semantic Wikis.* Paper presented at the Workshop on Semantic Web-enabled Software (Collocated with the International Semantic Web Conference), Galway, Ireland.

Decker, B., Ras, E., Rech, J., Klein, B., Reuschling, C., Höcht, C., et al. (2005). *A Framework for Agile Reuse in Software Engineering using Wiki Technology.* Paper presented at the Conference Professional Knowledge Management - Experiences and Visions., Kaiserslautern, Germany.

Avram, G., Ras, E., & Weibelzahl, S. (2004). *Using Weblogs for Eliciting new Experiences and Creating Learning Elements for Experienced-based Information Systems.* Paper presented at the 4th International Conference on Knowledge Management, Graz, Austria.

Ras, E., & Weibelzahl, S. (2004). *Embedding Experiences in Micro-didactical Arrangements.* Paper presented at the 6th International Work-

shop on Advances in Learning Software Organisations (LSO 2004), Banff, Canada.

Rech, J., & Ras, E. (2004). *Experience-Based Refactoring for Goal-Oriented Software Quality Improvement.* Paper presented at the First International Workshop on Software Quality (SOQUA 2004), Erfurt, Germany.

Rech, J., Ras, E., & Jedlitschka, A. (2004). Improving Software Quality through Refactoring by means of Didactical Augmented Experience. Paper presented at the *Conference Testing of Component-Based Systems and Software Quality*, vol. P-58. Lecture Notes in Informatics (LNI), Beydeda, S., Gruhn, V., Mayer J.D., Reussner, R., Schweiggert, F., Eds.: GI, pp. 141-149.

3. Book chapters

Memmel, M., Ras, E., Jantke, J. K., & Yacci, M. (2006). Approaches to Learning Object Oriented Instructional Design. In A. Koohang & K. Harman (Eds.), *Principles and Practices of the Effective Use of Learning Objects* (pp. 281-326). Santa Rosa, California: Informing Science Press.

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

4. Journal publications

Ras, E., & Rech, J. (2009). Using Wikis to Support the Net Generation in Improving Knowledge Acquisition in Capstone Projects. *Journal of Systems and Software*, 82, 553-562.

Ras, E., Carbon, R., Decker, B., & Rech, J. (2007). Experience management wikis for reflective practice in software capstone projects. *IEEE Transactions on Education, 50*(4), 312-320.

Decker, B., Ras, E., Rech, J., Jaubert, P., & Rieth, M. (2007). Wiki-Based Stakeholder Participation in Requirements Engineering. *IEEE Software, 24*(2), 28-35.

Rech, J., & Ras, E. (2007). Systematische Aggregation von Erfahrungen im Erfahrungsmanagement. *Künstliche Intelligenz, 21*(4), 16-22.

Rech, J., Ras, E., & Decker, B. (2007). Intelligent Assistance in German Software Development: A Survey. *IEEE Software, 24*(4), 72-79.

Ras, E., Rech, J., & Decker, B. (2007). Lernräume für erfahrungsbasiertes Lernen mit Wiki-Systemen im Software Engineering. *Zeitschrift für E-Learning - Lernkultur und Bildungstechnologie 2*(1, E-Learning und Wissensmanagement), 22-35.

Ras, E., Avram, G., Weibelzahl, S., & Waterson, P. (2005). Using Weblogs for Knowledge Sharing and Learning in Information Spaces. *Journal of Universal Computer Science, 11*(3), 394-409.

Ras, E. (2003). Life Long Learning. Modulare Wissensbasen für elektronische Lernumgebungen. *KI - Künstliche Intelligenz, 3,* 59.

5. Edited books and Proceedings

Rech, J., Decker, B., & Ras, E. (2008). *Emerging Technologies for Semantic Work Environments: Techniques, Methods, and Applications*. Hershey, New York, USA: Information Science Reference, IGI Global.

Ras, E., Memmel, M., Lindstaedt, S., Ley, T., & Albert, D. (2008). Proceedings of the Special Track on Intelligent Assistance for Self-Directed and Organizational Learning (IWL-LOKMOL '08), Graz, Austria.

Memmel, M., Ras, E., Wolpers, M., & Van Assche, F. (2007). Proceedings of the 3rd Workshop on Learner-Oriented Knowledge Management & KM-Oriented E-Learning (LOKMOL 2007), Crete, Greece.

Memmel, M., Ras, E., Weibelzahl, S., & Burgos, D. l. (2006). Proceeding of the Joint International Workshop on Professional Learning, Competence Development and Knowledge Management (LOKMOL and L3NCD), Crete, Greece.

Ras, E., Memmel, M., & Weibelzahl, S. (2005). Proceedings of the *First Workshop on Learner-Oriented Knowledge Management & KM-oriented E-Learning (LOKMOL 2005),* Kaiserslautern, Germany.

Appendix A. Material of the Experiment

This section contains all the questionnaires used during the experiment:

- Briefing questionnaire for assessing the disturbing factors related to experience and learning style (Appendix A.1)
- Pre- and post-test questionnaires for assessing the knowledge acquisition difference (Appendix A.2)
- Template of an experience package and two examples of experience packages that were used in the experiment (Appendix A.3)
- Assignments for assessing reading time, application time, efficiency, completeness, and accuracy (Appendix A.4)
- Debriefing questionnaire for assessing the other disturbing factors (Appendix A.5)

A.1. Briefing Questionnaire

Please answer the following questions. This will take you about 5 minutes. During the analysis of the data, the data will be anonymized – your name and Matr.-Nr. (enrollment no.) will be removed.

Subject-ID	**<the ID will be inserted by the evaluators>**
Name:	
Matr-Nr:	

Questions on University Education

<B1>	**Education**	
<B1.1>	Name of study (e.g., "Angewandte Informatik")	
<B1.2>	Major Subject (i.e., "Hauptfach/Vertiefung"):	
<B1.3>	Minor Subject (i.e., "Nebenfach/Wahlfach"): *(if more than one, please mention all)*	
<B1.4>	Which lectures regarding "Software Engineering" (e.g., "SE 1-3", "GSE", …) have you completed?	
<B1.5>	Number of terms (Fachsemester) completed (including the current one):	
<B1.6>	In how many practical courses (i.e., SE-oriented "Praktika") have you participated?	

Questions on Practical Software Engineering Experience

<B2>	Practical Software Engineering Experience	Yes	No
<B2.1>	Have you ever a written software system with more than 5 classes or 1000 lines of code?		
<B2.2>	Have you ever written software outside of university programs (e.g., private, commercial, OSS)?		
<B2.3>	Have you developed software in a large team (>4 persons) with distributed roles?		
<B2.4>	Have you developed software in a project with long duration (>6 months)?		

Questions on Experience with Programming & Java

<B3>	Questions on Experience with Programming & Java	
<B3.1>	How many years of computer programming experience do you have, if any?	
<B3.2>	How many different applications have you programmed?	
<B3.3>	How many different applications have you programmed in Java?	
<B3.4>	How many years were you involved in maintaining & improving a software system?	

<B4>	What is your experience with …	High Experience				No Experience		
<B4.1>	Java APIs (java.util, java.io, java.net, etc.)	○	○	○	○	○	○	○
<B4.2>	Java GUIs (AWT, Swing, SWT, etc.)	○	○	○	○	○	○	○
<B4.3>	Creating Java programs from scratch	○	○	○	○	○	○	○
<B4.4>	Debugging large Java programs	○	○	○	○	○	○	○
<B4.5>	The eclipse IDE (as a user, not plugin-developer)	○	○	○	○	○	○	○
<B4.6>	Other IDE such as Netbeans, Visual Studio, jBuilder, etc. (as a user, not plugin-developer)	○	○	○	○	○	○	○

Questions on Experience with Refactoring & Code Smells

<B5>	General Questions	
<B5.0>	Have you heard of refactoring before?	
<B5.1>	How many years of experience do you have with refactoring?	
<B5.2>	How many different applications have you refactored? (all programming languages)	
<B5.3>	How many different applications have you refactored in Java?	

<B6>	What is your practical experience with …	High Experience				No Experience		
<B6.1>	Identifying code smells, anti-patterns, pitfalls, design flaws, etc.	○	○	○	○	○	○	○
<B6.2>	Applying Refactorings manually	○	○	○	○	○	○	○
<B6.3>	Applying Refactorings such as “Extract Method” built into an IDE (except the “rename” refactoring)	○	○	○	○	○	○	○
<B6.4>	Working with design patterns, design heuristics, design principles, etc.	○	○	○	○	○	○	○

Questions on Experience with Software Quality Assurance & Maintenance

<B7>	What is your practical experience with …	High Experience						No Experience
<B7.1>	Quality models (such as ISO 9126, FURPS, Dromey, Boehm, …)	○	○	○	○	○	○	○
<B7.2>	Testing a software system?	○	○	○	○	○	○	○
<B7.3>	Inspecting a software system regarding quality issues?	○	○	○	○	○	○	○
<B7.4>	Software measurement (Metrics)?	○	○	○	○	○	○	○
<B7.5>	Code checking tools such as PMD, checkstyle, etc.?	○	○	○	○	○	○	○

<B8>	What is your practical experience with …	High Experience						No Experience
<B8.1>	Maintaining a software system? (e.g., managing defects, applying changes, etc.)	○	○	○	○	○	○	○
<B8.2>	Porting a software system to another platform? (e.g., Java 1.2 to 5.0, Java to C#, etc.)	○	○	○	○	○	○	○
<B8.3>	Improving a software system regarding efficiency (time behavior, resource behavior)?	○	○	○	○	○	○	○
<B8.4>	Improving a software system regarding reliability? (i.e., "Zuverlässigkeit")	○	○	○	○	○	○	○
<B8.5>	Improving a software system regarding usability?	○	○	○	○	○	○	○
<B8.6>	Improving a software system regarding functionality (suitability, interoperability, security)	○	○	○	○	○	○	○

Questions on Learning Style

<B9>	What is your most preferred learning style? (select one option)	
<B9.1>	Reading textbooks (with exercises)	○
<B9.2>	Classroom lectures (with exercises)	○
<B9.3>	Group work (interaction with peers and teacher / including exercises)	○
<B9.4>	Web-based training modules (with computer interaction / including examples and exercises)	○
<B9.5>	Trial and error approach (e.g., program, debug, repeat)	○

Thanks for filling out the questionnaire!

A.2. Pre- & Post-Questionnaires

This questionnaire serves to assess your competencies in the domain of refactoring and code smells. Please fill out the questionnaire as accurately as you can.
When you don't know the answer, please put your checkmark in the field "?" (Germ. Damit ist gemeint, dass Ihr nicht raten solltet – das würde die Ergebnisse verfälschen)
Before the data is processed, the data will be anonymized.
The results of the questionnaire have no impact on your grade (Germ. Note) of this practicum!

Subject-ID	**<this will be filled out by the evaluators>**
Name:	
Matr-Nr:	

General Understanding of Refactoring

<P1>	**What is refactoring about?**	**Yes**	**No**	**?**
<P1.1>	Refactoring transforms software in a way that it remains functionally identical	x		
<P1.2>	Refactoring is the art of safely removing the bad design decisions of existing code	x		
<P1.3>	Refactoring is rewriting code from scratch		x	
<P1.4>	Refactoring is dependent on eXtreme Programming (XP) methods		x	
<P1.5>	Refactoring is about a safe design-to-source transformation		x	
<P1.6>	Refactoring is about a safe source-to-source transformation	x		

<P2>	**What should be affected by refactoring?**	**Yes**	**No**	**?**
<P2.1>	The software's complexity	x		
<P2.2>	The software's flexibility	x		
<P2.3>	The software's understandability	x		
<P2.4>	The software's functionality		x	
<P2.5>	The behavior of the methods, classes, and components	x		
<P2.6>	The observable behavior of the software from the perspective of the user		x	
<P2.7>	The program's syntax	x		
<P2.8>	The software's performance	x		
<P2.9>	The program's semantics (meaning of methods, classes, etc.)	x		
<P2.10>	The program's size	x		

<P3>	When and how should a refactoring be considered?	Yes	No	?
<P3.1>	When a design choice is not explicitly addressed in one place in a system	x		
<P3.2>	When a code smell has been detected	x		
<P3.3>	When a system failure has been detected (e.g., by testing)		x	
<P3.4>	When the system design has a weakness	x		
<P3.5>	Refactoring is done on a periodical basis		x	
<P3.6>	Before implementing a new feature and if the design does not fit this change	x		
<P3.7>	Refactorings are always performed in small steps with compilation and test in-between	x		
<P3.8>	Refactorings are implemented completely. Afterwards, compilations and test are done because only completed refactorings result in a running system		x	
<P3.9>	Refactoring can be applied when the unit and acceptance tests haved failed; refactoring can help to solve the detected failures.		x	
<P3.10>	Refactoring should only be applied when the required automated unit or acceptance tests have been conducted successfully.	x		

<P4>	What are code smells?	Yes	No	?
<P4.1>	Code Smells are weaknesses in the requirements		x	
<P4.2>	Code Smells are failures observed by the user		x	
<P4.3>	Code Smells are defects observed by the tester		x	
<P4.4>	Code Smells are defects observed by the developer	x		
<P4.5>	Code Smells are weaknesses in the design	x		
<P4.6>	All Code Smells can be easily determined by using appropriate measures		x	
<P4.7>	Determining what is and is not a Code Smell is often a subjective judgment	x		

Assigment of Refactoring Methods to Code Smells

<P5>	**What refactorings are used to remove the following code smells?** **<place checkmarks in the columns for each code smell>** **<for those refactorings where you don't know which code smells they are suitable for, choose "?" >**								
		?	Comment	Long Method	Type Embedded in Name	Uncommunicative Name	Long Parameter List	Lazy Class	Data Class
<P5.1>	AddParameter						x		
<P5.2>	DecomposeConditional			x					
<P5.3>	EncapsulateCollection								x
<P5.4>	EncapsulateField								x
<P5.5>	ExtractMethod		x	x					
<P5.6>	HideMethod								
<P5.7>	IntroduceAssertion		x						
<P5.8>	IntroduceParameterObject			x			x		
<P5.9>	MoveMethod								x
<P5.10>	PreserveWholeObject			x			x		
<P5.11>	RemoveParameter								
<P5.12>	RemoveSettingMethod								
<P5.13>	RenameMethod		x		x	x			
<P5.14>	ReplaceMethodwithMethodObject			x					
<P5.15>	ReplaceParameterwithMethod						x		
<P5.16>	ReplaceTempwithQuery			x					

Questions related to the code smell *Long Method*

<C2>	**Questions related to the code smell *Long Method***	
<C2.1>	Explain in your own words what a *Long Method* code smell is? What are the problems it brings to the code?	**?**
	<Your answer:>	
<C2.2>	Mark the blocks in the following method that you would extract in order to make the method shorter (with your text marker)	

```
//example from Wakes p. 23
import java.util.*;
import java.io.*;

public class Report {
        public static void report(Writer out, List machines, Robot ro-
bot)
        throws IOException {
                out.write("FACTORY REPORT\n");
                out.write("This list includes information on
"+machines.size()+ " machines")
                Iterator line = machines.iterator);
                while (line.hasNext() {
                        Machine machine = (Machine) line.next();
                        out.write("Machine " + machine.name());
                        if (machine.status() != null)
                                out.write(" status=" + machine.status());
                        out.write("\n");
                }
                out.write("\n");

                out.write("Robot ");
                if (robot.location() != null)
                        out.write("location=" + ro-
bot.location().name()):
                if(robot.status() != null)
                        out.write("status=" + robot.status());

                out.write("\n");
                out.write("=========\n")
        }
}
```

C2.3>	Rewrite the report(…) method, as you have done the extract method for each block. (don't describe the new methods – only the new report() with the call of the extracted methods	**?**

<Your answer:>

```
public static void report (
Printstream out, L i s t machines. Robot robot) {
        reportHeader(out);
        reportMachines(out, machines);
        reportRobot(out, robot);
        reportFooter(out);
}
```

<C2.4>	What refactorings are suitable for the code smell *Long Method* in general? Name them all.	**?**
	<Your answer:> ExtractMethod IntroduceParameterObject PreserveWholeObject ReplaceTempWithQuery ReplaceMethodWithMethodObject	
<C2.5>	In what order should the previously listed refactorings be applied? <put "no sequence" if the sequence is not important>	**?**
	<Your answer:> 1. ExtractMethod 2. IntroduceParameterObject, 3. PreserveWholeObject, 4. ReplaceTempWithQuery 5. ReplaceMethodWithMethodObject	

<C2.6> What refactoring would you apply for this *Long Method* code smell example first? Please mark the code smell and explain why you apply this refactoring. **?**

```
class Customer ...
       public String statement() (
               double totalAmount = 0;
               int frequentRenterPoints = 0;
               Enumeration rentals = _rentals.elements();
               String result = "Rental Record for " + getName() + "\n";
               while (rentals.hasMoreElements()) {
                       Rental each = (Rental) rentals.nextElement();

                       //add frequent renter points
                       frequentRenterPoints ++;

                       //add bonus for a two day new release rental
                       if ((each.getMovie().getPriceCode() ==
Movie.NEW_RELEASE)
                                       && each.getDaysRented() > 1) {

                               frequentRenterPoints ++;
                       }
                       //show figures far this rental
                       result += "\t" + each.getMovie().getTitle()+
"\t" +
                       String.valueOf(each.getCharge()) + "\n";
                       totalAmount += each.getCharge();
               }

               //add footer lines
               result += "Amount owed is " +
String,valueOf(totalAmount) + "\n";
               result += "You earned " +
String.valueOf(frequentRenterPoint) + "frequent renter points";
               return result;
}
```

<Your answer:>

Use Extract Method

Questions related to the code smell *Type Embedded in Name*

<C3>	**Questions related to the code smell *Type Embedded in Name***			
<C3.1>	Explain in your own words what a *Type Embedded in Name* code smell is? What are the problems it brings to the code?	**?**		
	<Your answer:> The following problems are related to the code smell Type Embedded in Name. • Method names are compound words, consisting of a word plus the type of the argument(s).For example, a method addCourse(Course c). • Names are in Hungarian notation, where the type of an object is encoded into the name; e.g., icount as an integer member variable. • Variable names reflect their type rather than their purpose or role.			
<C3.2>	Which of the following examples included is a *Type Embedded in Name* code smell? <please mark the smell with your pen>	**Yes**	**No**	**?**

	`public Class getColumnClass(final int columnIndex) {`
	`  return String.class;`
	`}`
		x		

	`public class Texts {`
	`  private static final String BUNDLE_NAME =`
	`"de.frewert.dndinfo.gui.dndinfo"; //$NON-NLS-1$`
	
	`  private static final ResourceBundle RESOURCE_BUNDLE =`
	`  ResourceBundle.getBundle(BUNDLE_NAME);`
	
	`  private Texts()`
	
	`  public static String getString(String key) {`
	`  try {`
	`  return RESOURCE_BUNDLE.getString(key);`
	`  } catch (MissingResourceException e) {`
	`  return '!' + key + '!';`
	`  }`
	`  }`
	`}`
		x		

	`public void paintComponent(Graphics g) {`
	`  super.paintComponent(g);`
	
	`  Graphics2D g2 = (Graphics2D) g;`
	`  g2.setFont(bgFont);`
	`  g2.setColor(fontColor);`
	
	`  int dividerPos = getDividerLocation();`
	`  drawCentered(g2, info[0], 0, dividerPos);`
	`  drawCentered(g2, info[1], divider-`
	`  Pos + getDividerSize(),`
	`  getHeight());`
	`  }`
			x	

	`private Observer dndObserver = new Observer() { public void update(Observable o, Object arg) { if (arg instanceof DataFlavor[]) { gui.displayFlavors((DataFlavor[]) arg); } else if (arg instanceof String) { gui.appendData((String) arg); } else if (arg instanceof int[]) { int [] action = (int[]) arg; gui.setSourceActions(action[0]); gui.setUserAction(action[1]); } } }`	x		
	`private ActionListener quitListener = new ActionListener() { public void actionPerformed(ActionEvent e) { Main.this.quit(); } }`		x	
<C3.3>	Give another simple example of a code smell *Type Embedded in Name*		**?**	
	<Your answer:>			
<C3.4>	Please name the refactoring applied to the following *Type Embedded in Name* code smell:		**?**	
	<your answer:> RenameMethod			

	`public void storeTask (Task t) { t.setTaskId(numberTasks+1); currentTaskList.addTask(t); numberTasks++; }`
	
	is transformed to:
	
	`public void store (Task t) { t.setTaskId(numberTasks+1); currentTaskList.addTask(t); numberTasks++; }`			

<C3.5>	List the refactorings that are suitable for the code smell *Type Embedded in Name* in general		**?**	
	<Your answer:> RenameMethod			

<C3.6>	In what order should the previously listed refactorings be applied? <put "no sequence" if the sequence is not important>	**?**
	<Your answer:> RenameMethod	
<C3.7>	What refactoring would you apply for this *Type Embedded in Name* code smell example? Please mark each code smell with your text marker and explain why you apply this refactoring.	**?**

	`public void addDropTargetListener(DropTargetListener dtl) {`
	`/*`
	`* Using the GlassPane as only DropTarget would be more`
	`* elegant, but Drag&Drop doesn't work with a`
	`* GlassPane in Java <= 1.4.0. (See Java bug #4435403)`
	`*/`
	
	`// Use the following block if JRE 1.3 compatibility`
	`// isn't neccessary any longer.`
	
	`// Component c = SwingUtilities.getRoot(this);`
	`// if ((c != null) && (c instanceof JFrame)) {`
	`// JFrame f = (JFrame) c;`
	`// Component glassPane = f.getGlassPane();`
	`// glassPane.setVisible(true);`
	`// DropTarget dropTarget = new DropTarget(glassPane, dtl);`
	`// }`
	
	`new DropTarget(flavorArea, dtl);`
	`new DropTarget(dataArea, dtl);`
	`}`	

	<Your answer:> addDropTargetListener(DropTargetListener is a type embedded in name code smell. The variable type is embedded in the method name. When the type changes, the method also needs to be renamed.	

Questions related to the code smell *Comments*

<C4>	Questions related to the code smell *Comments*			
<C4.1>	Explain in your own words what a *Comments* code smell is? What are the problems if brings to the code?		?	
	<Your answer:> Comments should be used to give overviews of code and provide additional information that is not readily available in the code itself. Comments should contain only information that is relevant to reading and understanding the program and should be added when the author realizes that something isn't as clear as it could be and adds a comment. In addition, the frequency of comments sometimes reflects poor quality of code. A lot of comments can be reflected just as well in the code itself.			
<C4.2>	Which of the following examples includes at least one *Comments* code smell? <please mark the smell(s) with your text marker>	**Yes**	**No**	**?**
	`private JScrollPane getFlavorScrollPane(final Map map, String header1, String header2) { JTable table = new JTable(new FlavorTableModel(map, header1, header2)); final int viewportHeight = 12 * table.getRowHeight(); table.setPreferredScrollableViewportSize(new Dimension(450, viewportHeight)); // table.getColumn(header1).setPreferredWidth(header1.); JScrollPane scrollPane = new JScrollPane(table); scroll-Pane.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS); return scrollPane; }`	x		
	`public class AboutDialog extends JDialog { private static final long serialVersionUID = 3257853194578048567L; /** * Create a new AboutDialog. * @param parent the parent frame. * @param title the title of the dialog * @param version the version of the application */ public AboutDialog(Frame parent, String title, String ver-sion) { super(parent, Texts.getString("AboutDialog.title.prefix") + title); //$NON-NLS-1$ createGui(title, version); pack(); setResizable(false); }`	x		
	`/** * Liest die Refaktorierungen eines Diagnose-Plug-Ins aus der Extension- * Beschreibung aus.`		x	

	*
	* @param extensionID
	* ID der Extension, die den Extension-Point <co-
	de>Diagnosis</code>
	* implementiert
	* @return Refaktorierungen als Komma-separierte Liste in
	einem String
	*/
	public String getRefactorings(String extensionID) {
	return getAttributeValue(extensionID,
	EP_DIAGNOSIS,
	"refactorings", ELE-
	MENT_FRONTEND);
	}			

	private ActionListener buttonListener = new ActionListener()
	{
	public void actionPerformed(ActionEvent e) {
	// Don't dispose, dialog is reused in Main class
	FlavorDialog.this.~~hide~~();
	}
	};
		x		

	// Constructor where the Id is set
	public TaskList(int taskListId){
	this.taskListId =taskListId;
	state=false;
	tasks=new HashSet<Task>();
	}
		x		

<C4.3>	Give another simple example of a code smell *Comment*		?	
	<Your answer:>			
<C4.4>	Please name the refactoring applied to the following *Comments* code smell:		?	
	<your answer:> IntroduceAssertion			

	/**
	* @param clipLimit has to be larger than zero
	* @param delta has to have a positive value
	*/
	public boolean match(int[] expected, int[] actual, int
	clipLimit, int delta)
	{
	// Clip " too- large" values
	for (int i = 0; i < actual.length; i++)
	if (actual [i] > clipLimit)
	actual [i] = clipLimit;
	// Check for length differences
	if (actual.length != expected.length)
	return false;
	// Check that each entry within expected +/-
	delta
	for (int i = 0; i < actual.length; i++)
	if (Math.abs(expected[i] - actual[i] >
	delta)
	return false;			

```
                        return true;
                                }

                        is transformed to:

public boolean match(int[] expected, int[] actual, int
clipLimit, int delta)
        {
                assert expected != null;
                assert actual != null;
                assert clipLimit >= 0;
                assert delta >= 0;

                // Clip " too- large" values
                for (int i = 0; i < actual.length; i++)
                        if (actual [i] > clipLimit)
                                actual [i] = clipLimit;
                // Check for length differences
                if (actual.length != expected.length)
                        return false;
                // Check that each entry within expected +/-
delta
                for (int i = 0; i < actual.length; i++)
                        if (Math.abs(expected[i] - actual[i] >
delta)
                                        return false;

                        return true;
        }
```

<C4.5>	List the refactorings that are suitable for the code smell *Comments* in general	**?**
	<Your answer:> ExtractMethod IntroduceAssertion RenameMethod	
<C4.6>	In what order should the previously listed refactorings be applied? <put "no sequence" if the sequence is not important>	**?**
	<Your answer:> doesn't matter, depends on the type of the *comments code smell*.	
<C4.7>	What refactoring(s) would you apply for this(these) *Comments* code smell example(s)? Mark each code smell with your text marker and explain why you apply this refactoring.	**?**

```
/** Simulation of a Tic-Tac-Toe game (does not do strategy).
*/
public class TicTacToe {
 protected static final int X = 1, O = -1;   // players
 protected static final int EMPTY = 0;              // empty
cell
 protected int board[][] = new int[3][3];    // game board
 protected int player;                       // current
player
 /** Constructor */
 public TicTacToe() { clearBoard(); }
 /** Clears the board */
 public void clearBoard() {
```

```
 for (int i = 0; i < 3; i++)
 for (int j = 0; j < 3; j++)
 board[i][j] = EMPTY;          // every cell should be empty
 player = X;                          // the first player is
'X'
 }
 /** Puts an X or O mark at position i,j */
 public void putMark(int i, int j) throws IllegalArgumentEx-
ception {
 if ((i < 0) { (i > 2) { (j < 0) { (j > 2))
 throw new IllegalArgumentException("Invalid board posi-
tion");
 if (board[i][j] != EMPTY)
 throw new IllegalArgumentException("Board position occu-
pied");
 board[i][j] = player;         // place the mark for the cur-
rent player
 player = - player;            // switch players (uses fact
that O = - X)
 }
 /** Checks whether the board configuration is a win for the
given player */
 public boolean isWin(int mark) {
 return ((board[0][0] + board[0][1] + board[0][2] == mark*3)
// row 0
         { (board[1][0] + board[1][1] + board[1][2] ==
mark*3) // row 1
         { (board[2][0] + board[2][1] + board[2][2] ==
mark*3) // row 2
         { (board[0][0] + board[1][0] + board[2][0] ==
mark*3) // column 0
         { (board[0][1] + board[1][1] + board[2][1] ==
mark*3) // column 1
         { (board[0][2] + board[1][2] + board[2][2] ==
mark*3) // column 2
         { (board[0][0] + board[1][1] + board[2][2] ==
mark*3) // diagonal
         { (board[2][0] + board[1][1] + board[0][2] ==
mark*3)); // diagonal
 }
 /** Returns the winning player or 0 to indicate a tie */
 public int winner() {
 if (isWin(X))
 return(X);
 else if (isWin(O))
 return(O);
 else
 return(0);
 }
```

<Your answer:>

It is clear that the constructor is the constructor!

The name of the method clearBoard tells the reader what the method does. The comment is redundant. The same is true for the methods putMark and isWin.

Questions related to the code smell *Uncommunicative Name*

<C4>	Questions related to the code smell *Uncommunicative Name*			
<C4.1>	Explain in your own words what a *Uncommunicative Name* code smell is? What are the problems it brings to the code?		?	
	<Your answer:> A name doesn't communicate its intent of a method, variable, classes, etc. well enough - One- or two-character names - Names with vowels omitted - Numbered variables (e.g., panel, pane2, and so on) - Odd abbreviations - Misleading names			
<C4.2>	Which of the following examples includes at least one *Uncommunicative Name* code smell? <please mark the smell(s) with your text marker>	**Yes**	**No**	**?**

	`public Class getColumnClass(final int columnIndex) {`
	`  return String.class;`
	`  }`
			x	

	`public void addDropTargetListener(DropTargetListener dtl)`
	`{`
	`  new DropTarget(flavorArea, dtl);`
	`  new DropTarget(dataArea, dtl);`
	`  }`
		x		

	`public String getColumnName(final int column) {`
	`  String name = (column >= columnHeader.length)`
	`  ? ""`
	`  : columnHeader[column];`
	`  return (name == null) ? "" : name;`
	`  }`
			x	

	`public void insertUpdate(DocumentEvent e) {`
	`  /* using invokeLater seems neccessary */`
	`  SwingUtilities.invokeLater(new Runnable() {`
	`  public void run() {`
	`  scrollbar.setValue(scrollbar.getMaximum());`
	`  }`
	`  });`
	`  }`
		x		

	`public void paintComponent(Graphics g) {`
	`  super.paintComponent(g);`
	
	`  Graphics2D g2 = (Graphics2D) g;`
	`  g2.setFont(bgFont);`
	`  g2.setColor(fontColor);`
	
	`  int dividerPos = getDividerLocation();`
	`  drawCentered(g2, info[0], 0, dividerPos);`
	`  drawCentered(g2, info[1],`
	`  dividerPos + getDividerSize(), getHeight());`
	`}`
		x		

	`public Object getValueAt(final int arg0, final int arg1)`
	`{`
	`  return data[arg0][arg1];`
	`  }`
		x		

<C4.3>	Give another simple example of a *Uncommunicative Name* code smell	**?**
	<Your answer:>	
<C4.4>	Please name the refactoring applied to the following *Uncommunicative Name* code smell:	**?**
	<your answer:> RenameMethod (or RenameVariable)	

	`//data contains the colortable`
	
	`public Object getValueAt(final int arg0, final int arg1)`
	`{`
	`return data[arg0][arg1];`
	`}`
	
	is transformed to:
	
	`public Object getValueAt(final int x, final int y) {`
	`return data[x][y];`
	`}`	

<C4.5>	List the refactorings that are suitable for the code smell *Uncommunicative Name* in general	**?**
	<Your answer:> RenameMethod (or RenameVariable)	
<C4.6>	In what order should the previously listed refactorings be applied? <put "no sequence" if the sequence is not important>	**?**
	<Your answer:> RenameMethod (or RenameVariable)	
<C4.7>	What refactoring would you apply for this(these) *Uncommunicative Name* code smell example(s)? Mark the code smell with your text marker and explain why you apply this refactoring.	**?**

	`private Observer dndObserver = new Observer() {`
	`public void update(Observable o, Object arg) {`
	`if (arg instanceof DataFlavor[]) {`
	
	`gui.displayFlavors((DataFlavor[]) arg);`
	
	`} else if (arg instanceof String) {`
	
	`gui.appendData((String) arg);`
	
	`} else if (arg instanceof int[]) {`
	
	`int [] action = (int[]) arg;`
	`gui.setSourceActions(action[0]);`
	`gui.setUserAction(action[1]);`
	`}`
	`}`
	`};`	

	<Your answer:> o is a one-character variable arg is a variable name with no meaning	

A.3. Experience Packages for Experimentation

A.3.1 Experience Package Template

Titel of EP			**Type**	Experience
Action (A)				
	Abstract:			
	Problem:			
	Solution:			
Benefit (B)				
	Effect:			
Context (C)				
	Product:			
	Process:			
	Project:			
	Knowledge:			
	Organization:			
	People:			
	Group:			
Description (D)				
	Explanation:			
	Example:			
Evidence (E)	Analysis Technique:		Hypothesis:	
Administrative				
	Author:		Date:	
	Version:		Relation EPs:	
	Status:			
Remark				

A.3.2 Experience Package: Code Smell *Long Method*

Titel of EP	Code Smell Long Method	**Type**	Experience
Action (A)			
	Abstract:	Large methods consist of a large number of lines. You should be suspicious when a method has more than 5 to 10 lines. The refactorings ExtractMethod, ReplaceTempwithQuery, ReplaceMethodwithMethodObject, DecomposeConditional can be used to reduce this kind of code smell. They will improve the class structure and abstraction levels.	
	Problem:	A method starts down a path and, rather than break the flow or identify the helper classes, the author adds more and more. Code is often easier to write than it is to read, so there's a temptation to write blocks that are too big, which means that they get difficult to maintain, understand, etc.	
	Solution:	The refactoring *ExtractMethod* could be used to break up the method into smaller parts. Look for comments or white space delineating interesting blocks. You want to extract methods that are semantically meaningful, not just introduce a function call every seven lines. In addition, the following three methods can be used, too: - *ReplaceTempwithQuery*: Temporary variables are used to hold the result of an expression. This expression should be replaced with a method. Extract the expression into a method. - *ReplaceMethodwithMethodObject*: The difficulty in decomposing a method lies in local variables. If they are rampant (Germ. üppig), decomposition can be difficult. Applying it turns all the local variables into fields on the method object and ExtractMethod can be applied on this new object afterwards. - *DecomposeConditional*: Methods named after the intention of that block of code replace the parts of the conditional part and each of the alternatives. This way you highlight the condition and make it clear what you are branching on.	
Benefit (B)			
	Effect:	Improves communication. May expose duplication. Often helps to get new classes and abstractions	

Context (C)				
	Product:	Java Code		
	Process:	ExtractMethod, ReplaceTempwithQuery, ReplaceMethodwithMethodObject, DecomposeConditional		
	Project:	OO projects		
	Knowledge:	Code Smell Long Method		
	Organization:	Fraunhofer IESE		
	Individual:	Eric Ras		
	Group:	SOP-Dev		
Evidence (E)	Analysis Technique:	-	Hypothesis	-
Administrative				
	Author:	Martin Fowler	Date:	1999
	Version:		Relation EPs:	
	Status:			
remark	Exercise	W. Exercise 4 23ff		

A.3.3 Experience Package: Code Smell *Type Embedded in Name*

Titel of EP	Code Smell Type Embedded in Name	**Type**	Experience
Action (A)			
	Abstract:	When types are embedded in names, it's not only redundant, but it forces you to change the name if the type changes. This often results. Therefore, the refactoring RenameMethod is applied to avoid this kind of code smell, which is called Type Embedded in Name. Avoid placing types in method names!	
	Problem:	The embedded name can create unnecessary troubles because later changes of the parameter (i.e., type) will lead to a renaming of the method and the related calls.	
	Solution:	The refactoring *RenameMethod* (the same is done for fields or constants) should be applied, which leads to a new name that communicates the intent of the method without being so much tied to a type.	
Benefit (B)			
	Effect:	Improves communication. May make it easier to spot duplication.	

Context (C)				
	Product:	Java code		
	Process:	RenameMethod		
	Project:	OO projects		
	Knowledge:	Code Smell Type Embedded in Name		
	Organization:	Fraunhofer IESE		
	Individual:	Eric Ras		
	Group:	Sop-Dev		
Evidence (E)	Analysis Technique:		Hypothesis	
Administrative				
	Author:	Wakes	Date:	
	Version:	1.0	Relation EPs:	
	Status:	stable		
remark	exercise	no one in W.		

A.4. Assignments

In the following, one example of an assignment for the experimental group (Day 1) is given to show how an assignment is structured and to illustrate how the data was gathered during the experiment. Five different developer teams were involved during the experiment. The code used for the assignments was code produced by the corresponding teams themselves, i.e., the assignments contain their own code. Therefore, 20 different assignments were produced for the two periods of the experiment. Due to space limitations in this work, the complete set of assignments can be found in a technical report (Ras, 2009b) (ca. 160 pages).

The first page of the assignment provides instructions for solving the assignment and asks the subject to enter the time when he starts to solve the assignment (see Appendix A.4.1). After that, two exercises with Java code were given to the subjects (see Appendix A.4.2 and Appendix A.4.3). It was up to the students to decide whether they completely read the provided information first (i.e., information of an experience package or a learning space) or directly started to solve the exercises. The sheet for describing the solutions used by the subjects is provided in Appendix A.4.4 (example) and Appendix A.4.5 (empty sheet).

A.4.1 Assignment Information and Related Exercises (Mo-Mo-G1):

(Group:_____)
Your Name: _____
Your Subject-ID: _____<your ID will be filled out by evaluators>

Goal of the experiment:
The goal of the experiment is to apply the knowledge from an experience package to your own context (in this case, the DCGA project). Information about the experience package will be provided in a Wiki. Further, additional information in a so-called learning space will help you to understand and apply the experience package. In order to apply the experience packages, an exercise should be solved.

Selected Experience Packages
This sheet explains in which order you should work through the experience packages. Two experience packages have been assigned to you. Please access them in the following sequence as assigned in the parentheses. When you have read the information in the Wiki and when you think you are ready to solve the exercise, please put the actual time behind the corresponding experience package when you start to access the experience package in the Wiki.

- Experience Package Code Smell *Comments* (___)
starting time [___ : ___];
- Experience Package Code Smell *Long Method* (___)
starting time [___ : ___]
- Experience Package Code Smell *Type Embedded in Name* (___)
starting time [___ : ___]
- Experience Package Code Smell *Uncommunicative Name* (___)
starting time [___ : ___]
- Experience Package Code Smell *Long Parameter List* (___)
starting time [___ : ___] ending time [___ : ___];
- Experience Package Code Smell *Lazy Class* (___)
starting time [___ : ___] ending time [___ : ___];
- Experience Package Code Smell *Data Class* (___)
starting time [___ : ___] ending time [___ : ___];

Please access the Wiki by using your web browser:
http://watt.informatik.uni-kl.de/gseprojekt1/index.php/Spezial:Experiences
(use gseprojekt "1" !)
Login: experiment
Passwd: geiermeier

The exercises are provided in the following.

A.4.2 Exercise to Experience Package for Amica Interaction Group: Long Method Code Smell

Your Name: _____
Your Subject-ID: _____<your ID will be filled out by evaluators>

Please put the starting time in here [___ : ___]
Please put the ending time in here [___ : ___]
Exercise:

1. **Identify and mark** with a text marker code smells of the following type: Long Method
2. For each identified code smell, **state the refactoring** you would apply to the code and **give a sequential number** - start with "1"
3. **Use the *Answer Sheet for Exercises*. Put the related number in the first column** in order to relate your answer to the identified code smell. Then **explain your decision** (i.e., your stepwise solution in your own words or why you wouldn't remove the code smell).

Amica_Interaction:match.java

```
package org.belami.dcga.amica_interaction.mapping;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import org.belami.dcga.amica_interaction.Situation;
import org.belami.dcga.common_datastructures.Information;
import org.belami.dcga.common_datastructures.Task;
import org.belami.dcga.common_datastructures.TaskEvent;
import org.w3c.dom.DOMException;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

/* Data structure containing the information of one "match"
element from the XML mapping file.
 */
public class Match {
 /**
* Fact ID that has to be matched with the Situation object
 */
 private String factName = null;
 /**
 * Comparator method for the fact ID from the mapping-file*/
```

<The complete Java code can be found in (Ras, 2009b)>

A.4.3 Exercise to Experience Package for Amica Interaction Group: Type Embedded in Name Code Smell

Your Name: _____

Your Subject-ID: _____<your ID will be filled out by evaluators>

Please put the starting time in here [___ : ___]
Please put the ending time in here [___ : ___]
Exercise:

1. **Identify and mark** with a text marker code smells of the following type: Type embedded in name
2. For each identified code smell, **state the refactoring** you would apply into the code and **give a sequential number** - start with "1"
3. **Use the *Answer Sheet for Exercises*. Put the related number in the first column** in order to relate your answer to the identified code smell. Then **explain your decision** (i.e., your stepwise solution in your own words or why you wouldn't remove the code smell).

Amica_Interaction:match.java

```
package org.belami.dcga.amica_interaction.mapping;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import org.belami.dcga.amica_interaction.Situation;
import org.belami.dcga.common_datastructures.Information;
import org.belami.dcga.common_datastructures.Task;

/* Data structure containing the information of one "match"
element from the XML mapping file.
 *
 */
public class Match {
 /**
* Fact ID that has to be matched with the Situation object
 */
private String factName = null;
 /**
 * Comparator method for the fact ID from the mapping-file
  */
```

<The complete Java code can be found in (Ras, 2009b)>

A.4.4 Answer Sheet for Exercises (example)

This is an example at how to mark a code smell and how to describe it in the *Answer Sheet for Exercises*.

Code example:

```
void printOwing() {
            printBanner();

            //print details
            System.out.println ("name:    " + _name);
            System.out.println ("amount   " + amount);
}
```

Your explanation can be provided in different ways:

Number	Explanation of your descision

1	I would use the Extract Method refactoring. This is a solution:
	`void printOwing() {`
	`  printBanner();`
	`  printDetails(getOutstanding());`
	`}`
	
	`void printDetails (double outstanding) {`
	`  System.out.println ("name:  " + _name);`
	`  System.out.println ("amount  " + outstanding);`
	`}`

	or describe it in this way
1	I would use the Extract Method refactoring. The first step is to extract both system.out.println statements into a separate method (e.g., method printDetails(double outstanding) with the double variable oustanding). This method call to this new method will replace the println statements in the printOwning method. That's it.
	It is not necessary to state the compile and test steps !

A.4.5 Answer Sheet for Assignments

The answer can also be stated in German if this is more appropriate for you.

Number	Explanation of your decision

A.5. Debriefing Questionnaire

Questions on Complexity of the Tasks

<D1>		Agree				Disagree		
<D1.1>	The complexity of the experience packages used in both runs (Monday and Tuesday) were comparable	○	○	○	○	○	○	○
<D1.2>	The complexity of the code in the exercises used in both runs (Monday and Tuesday) were comparable	○	○	○	○	○	○	○
<D1.3>	I knew most of the code in the exercises during both runs	○	○	○	○	○	○	○

Questions on Time Needed

<D2>	I had enough time to	Yes	No
<D2.1>	read the information provided by the learning spaces in run 1 (Monday)		
<D2.2>	read the information provided by the learning spaces in run 2 (Tuesday)		
<D2.3>	solve the exercises in run 1 (Monday)		
<D2.4>	solve the exercises in run 2 (Tuesday)		
<D2.5>	familiarize myself with the Wiki and the learning space		

Questions on Learning Spaces

These questions are related to the run where you had access to the Learning Space.

<D3>	How did you use the Learning Space (LS)? <choose one option>	
<D3.1>	I first read the LS completely and started to solve the exercises without accessing the LS again	○
<D3.2>	I first read the LS completely and started to solve the exercises by accessing the LS again	○
<D3.3>	I first read the LS partially and started to solve the exercises without accessing the LS again	○
<D3.4>	I first read the LS partially and started to solve the exercises by accessing the LS again	○
<D3.5>	I didn't read the LS and started with the exercise without accessing the LS at all	○
<D3.6>	I didn't read the LS and started with the exercise by accessing the LS later	○

<D4>	What kind of information did you find useful in the Learning Space with regard to solving the exercise?	Agree				Disagree		
<D4.1>	Descriptions of items → labeled as *Description*	○	○	○	○	○	○	○
<D4.2>	Definitions of items → labeled as *Definition*	○	○	○	○	○	○	○
<D4.3>	Example descriptions of items → labeled as *Example*	○	○	○	○	○	○	○
<D4.4>	Counterexample descriptions of items → labeled as *Counterexample*	○	○	○	○	○	○	○
<D4.5>	Process descriptions of items → labeled as *Process*	○	○	○	○	○	○	○

Questions on Stand-Alone Experience Package vs. Learning Spaces

Below you will find a number of opposing adjectives on both sides of each line. You can react to the statements by checking the appropriate point on the line, as in this example:

Useful				Useless		
●	○	○	○	○	○	○

when you think that it was very useful.

<D5>	useful				useless		
I consider the explanations / information provided in a Learning Space in addition to an experience package descrption in general	○	○	○	○	○	○	○
	boring				absorbing		
	○	○	○	○	○	○	○
	easy				difficult		
	○	○	○	○	○	○	○
	clear				confusing		
	○	○	○	○	○	○	○
	complete				incomplete		
	○	○	○	○	○	○	○

<D6>	useful				useless		
I consider the explanations / information provided in a stand-alone experience package description (without Learning Space) in general	○	○	○	○	○	○	○
	boring				absorbing		
	○	○	○	○	○	○	○
	easy				difficult		
	○	○	○	○	○	○	○
	clear				confusing		
	○	○	○	○	○	○	○
	complete				incomplete		
	○	○	○	○	○	○	○

<D7> I would like to make the following comment(s) / improvement suggestion(s) (can be in German)

<D8> I had a problem with … <please explain (can be in German)>:

Questions on Evaluating of the Use and Acceptance of Learning Spaces

<D9> Performance expectancy		**Agree**				**Disagree**		
<D9.1>	I would find the system useful in my job.	○	○	○	○	○	○	○
<D9.2>	Using the Learning Space enables me to accomplish tasks more quickly.	○	○	○	○	○	○	○
<D9.3>	Using the Learning Space increases my productivity.	○	○	○	○	○	○	○
<D9.4>	If I use the Learning Space, I will increase my chances of getting a pay raise.	○	○	○	○	○	○	○

<D10> Effort expectancy		**Agree**				**Disagree**		
<D10.1>	My interaction with the Learning Space would be clear and understandable.	○	○	○	○	○	○	○
<D10.2>	It would be easy for me to become skillful at using the Learning Space.	○	○	○	○	○	○	○
<D10.3>	I would find the Learning Space easy to use.	○	○	○	○	○	○	○

<D11> Attitude toward using technology		**Agree**				**Disagree**		
<D11.1>	Using the Learning Space is a good idea.	○	○	○	○	○	○	○
<D11.2>	The Learning Space makes work more interesting.	○	○	○	○	○	○	○
<D11.3>	Working with the Learning Space is fun.	○	○	○	○	○	○	○
<D11.4>	I like working with the Learning Space.	○	○	○	○	○	○	○

Thanks for filling out the questionnaire!

Appendix B. Material of the Case Study

Evaluierung des Learning Space Tools

Name: _____

(Der Name wird nur für Nachfragen bei Unklarheiten der Antworten benötigt. Der Fragebogen wird natürlich anonym ausgewertet.)

Zugangsdaten zum Server:

http://ls.sop-world.org/

Login: lernen

Passwort: lsdevserver

Dann bitte als Benutzer „test“ mit dem Passwort „erfahrung“ rechts oben im Wiki Fenster einloggen

I. ISONORM Fragebogen zur Software Ergonomie

Füllen Sie bitte den nachfolgenden Fragebogen aus. Die Fragen, die Ihrer Meinung nach nicht für dieses System zutreffen, lassen Sie bitte unbeantwortet.

Der Fragebogen entspricht dem ISONORM 9142/10 Fragebogen.

Die folgenden Fragen beziehen sich ausschließlich auf die Arbeitsaufgabe der Wiederverwendung von Erfahrung und der Verwendung von Lernräumen und nicht auf die anderen Wiki-Funktionalitäten.

Aufgabenangemessenheit

<E1>	**Unterstützt die Software die Erledigung Ihrer Arbeitsaufgaben (Wiederverwendung von Erfahrung), ohne Sie als Benutzer unnötig zu belasten?**								
	Die Software...	---	--	-	-/+	+	++	+++	
<E1.1>	ist kompliziert zu bedienen.	○	○	○	○	○	○	○	ist unkompliziert zu bedienen.
<E1.2>	bietet nicht alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.	○	○	○	○	○	○	○	bietet alle Funktionen, die anfallenden Aufgaben effizient zu bewältigen.
<E1.4>	erfordert überflüssige Eingaben.	○	○	○	○	○	○	○	erfordert keine überflüssigen Eingaben.
<E1.5>	ist schlecht auf die Anforderungen der Arbeit zugeschnitten.	○	○	○	○	○	○	○	ist gut auf die Anforderungen der Arbeit zugeschnitten.

Selbstbeschreibungsfähigkeit

<E2>	**Gibt Ihnen die Software genügend Erläuterungen und ist sie in ausreichendem Maße verständlich?**								
	Die Software...	---	--	-	-/+	+	++	+++	
<E2.1>	bietet einen schlechten Überblick über ihr Funktionsangebot.	○	○	○	○	○	○	○	bietet einen guten Überblick über ihr Funktionsangebot.
<E2.2>	verwendet schlecht verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs.	○	○	○	○	○	○	○	verwendet gut verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs.
<E2.3>	liefert in unzureichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind.	○	○	○	○	○	○	○	liefert in zureichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind.

Steuerbarkeit

<E3>	**Können Sie als Benutzer die Art und Weise, wie Sie mit der Software arbeiten, beeinflussen?**								
	Die Software...	---	--	-	-/+	+	++	+++	
<E3.1>	bietet keine Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen.	○	○	○	○	○	○	○	bietet die Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen.
<E3.2>	erzwingt eine unnötig starre Einhaltung von Bearbeitungsschritten.	○	○	○	○	○	○	○	erzwingt keine unnötig starre Einhaltung von Bearbeitungsschritten.
<E3.3>	ermöglicht keinen leichten Wechsel zwischen einzelnen Menüs oder Masken.	○	○	○	○	○	○	○	ermöglicht einen leichten Wechsel zwischen einzelnen Menüs oder Masken.
<E3.4>	ist so gestaltet, dass der Benutzer nicht beeinflussen kann, wie und welche Informationen am Bildschirm dargeboten werden.	○	○	○	○	○	○	○	ist so gestaltet, dass der Benutzer beeinflussen kann, wie und welche Informationen am Bildschirm dargeboten werden.
<E3.5>	erzwingt unnötige Unterbrechungen der Arbeit.	○	○	○	○	○	○	○	erzwingt keine unnötigen Unterbrechungen der Arbeit.

Erwartungskonformität

<E4>	**Kommt die Software durch eine einheitliche und verständliche Gestaltung Ihren Erwartungen und Gewohnheiten entgegen?**								
	Die Software...	---	--	-	-/+	+	++	+++	
<E4.1>	erschwert die Orientierung, durch eine uneinheitliche Gestaltung.	○	○	○	○	○	○	○	erleichtert die Orientierung, durch eine einheitliche Gestaltung.
<E4.2>	lässt einen im Unklaren darüber, ob eine Eingabe erfolgreich war oder nicht.	○	○	○	○	○	○	○	lässt einen nicht im Unklaren darüber, ob eine Eingabe erfolgreich war oder nicht.
<E4.3>	informiert in unzureichendem Maße über das, was sie gerade macht.	○	○	○	○	○	○	○	informiert in ausreichendem Maße über das, was sie gerade macht.
<E4.4>	reagiert mit schwer vorhersehbaren Bearbeitungszeiten.	○	○	○	○	○	○	○	reagiert mit gut vorhersehbaren Bearbeitungszeiten.
<E4.5>	lässt sich nicht durchgehend nach einem einheitlichen Prinzip bedienen.	○	○	○	○	○	○	○	lässt sich durchgehend nach einem einheitlichen Prinzip bedienen.

Individualisierbarkeit

<E6>	Können Sie als Benutzer die Software ohne großen Aufwand an Ihre individuellen Bedürfnisse und Anforderungen anpassen?								
	Die Software...	---	--	-	-/+	+	++	+++	
<E6.2>	lässt sich von dem Benutzer schlecht an seine persönliche, individuelle Art der Arbeitserledigung anpassen.	○	○	○	○	○	○	○	lässt sich von dem Benutzer gut an seine persönliche, individuelle Art der Arbeitserledigung anpassen.
<E6.3>	eignet sich für Anfänger und Experten nicht gleichermaßen, weil der Benutzer sie nur schwer an seinen Kenntnisstand anpassen kann.	○	○	○	○	○	○	○	eignet sich für Anfänger und Experten gleichermaßen, weil der Benutzer sie leicht an seinen Kenntnisstand anpassen kann.
<E6.4>	lässt sich - im Rahmen ihres Leistungsumfangs - von dem Benutzer schlecht für unterschiedliche Aufgaben passend einrichten.	○	○	○	○	○	○	○	lässt sich - im Rahmen ihres Leistungsumfangs - von dem Benutzer gut für unterschiedliche Aufgaben passend einrichten.
<E6.5>	ist so gestaltet, dass der Benutzer die Bildschirmdarstellung schlecht an seine individuellen Bedürfnisse anpassen kann.	○	○	○	○	○	○	○	ist so gestaltet, dass der Benutzer die Bildschirmdarstellung gut an seine individuellen Bedürfnisse anpassen kann.

Lernförderlichkeit

<E7>	Ist die Software so gestaltet, dass Sie sich ohne großen Aufwand in sie einarbeiten konnten und bietet sie auch dann Unterstützung, wenn Sie neue Funktionen lernen möchten?								
	Die Software...	---	--	-	-/+	+	++	+++	
<E7.1>	erfordert viel Zeit zum Erlernen.	○	○	○	○	○	○	○	erfordert wenig Zeit zum Erlernen.
<E7.2>	ermutigt nicht dazu, auch neue Funktionen auszuprobieren.	○	○	○	○	○	○	○	ermutigt dazu, auch neue Funktionen auszuprobieren.
<E7.3>	erfordert, dass man sich viele Details merken muss.	○	○	○	○	○	○	○	erfordert nicht, dass man sich viele Details merken muss.
<E7.4>	ist so gestaltet, dass sich einmal Gelerntes schlecht einprägt.	○	○	○	○	○	○	○	ist so gestaltet, dass sich einmal Gelerntes gut einprägt.
<E7.5>	ist schlecht ohne fremde Hilfe oder Handbuch erlernbar.	○	○	○	○	○	○	○	ist gut ohne fremde Hilfe oder Handbuch erlernbar.

II. UTAUT Fragebogen zur Nutzung und Akzeptanz (in Englisch)

The following questions are based on the UTAUT (Unified Theory of Acceptance and Use of Technology).

<U1> Performance expectancy	**Agree**				**Disagree**		
<U1.1> I would find the system useful in my job.	○	○	○	○	○	○	○
<U1.2> Using the system enables me to accomplish tasks more quickly.	○	○	○	○	○	○	○
<U1.3> Using the system increases my productivity.	○	○	○	○	○	○	○
<U1.4> If I use the system, I will increase my chances of getting a pay raise.	○	○	○	○	○	○	○

<U2> Effort expectancy	**Agree**				**Disagree**		
<U2.1> My interaction with the system would be clear and understandable.	○	○	○	○	○	○	○
<U2.2> It would be easy for me to become skillful at using the system.	○	○	○	○	○	○	○
<U2.3> I would find the system easy to use.	○	○	○	○	○	○	○
<U2.4> Learning to operate the system is easy for me.	○	○	○	○	○	○	○

<U3> Attitude toward using technology	**Agree**				**Disagree**		
<U3.1> Using the system is a good idea.	○	○	○	○	○	○	○
<U3.2> The system makes work more interesting.	○	○	○	○	○	○	○
<U3.3> Working with the system is fun.	○	○	○	○	○	○	○
<U3.4> I like working with the system.	○	○	○	○	○	○	○

<U4> Facilitating conditions	**Agree**				**Disagree**		
<U4.1> I have the resources necessary to use the system.	○	○	○	○	○	○	○
<U4.2> I have the knowledge necessary to use the system.	○	○	○	○	○	○	○
<U4.3> The system is not compatible with other systems I use.	○	○	○	○	○	○	○
<U4.4> A specific person (or group) is available for assistance with system difficulties.	○	○	○	○	○	○	○

<U5> Self-efficacy	**Agree**				**Disagree**		
<U5.1> I could complete a job or task using the system…	○	○	○	○	○	○	○
<U5.2> If there was no one around to tell me what to do as I go.	○	○	○	○	○	○	○
<U5.3> If I could call someone for help if I got stuck.	○	○	○	○	○	○	○
<U5.4> If I had a lot of time to complete the job for which the software was provided.	○	○	○	○	○	○	○
<U5.5> If I had just the built-in help facility for assistance.	○	○	○	○	○	○	○

III. Weitere Anmerkungen, Kritik, Verbesserungsvorschläge …

… zur Farbgebung, Strukturierung der Informationen, Navigation

… zur Anreicherung von Erfahrungen mit Lernelementen (Integration von Wissensmanagement und E-Learning)

… zu Lernelementen

…

Appendix C. Additional Statistics

C.1. Principal Component Analysis of the Briefing Questionnaire

A principal component analysis (PCA) was performed for the briefing questionnaire for the purpose of data reduction and scale building.

In the following, only results from the PCA will be provided, since the process of the PCA is incremental and is based on many small analysis steps. Detailed explanations about the PCA can be found in (Bühner, 2006). The PCA was performed with varimax rotation. Components (i.e., experience factors) were only extracted when their eigenvalue passed 1.0, which means that the factor explains a higher variance than each single item that is part of the factor. The minimal *Kaiser-Meyer-Olkin coefficient* (KMO is a quality measure to measure whether the whole set of items is suitable for a PCA) of 0.6 has to be reached and a minimal *Measure of Sample Adequacy* (MSA is a quality measure to determine whether a single item is suitable for a PCA) is higher than 0.6 and is part of the anti-image correlation matrix. The Bartlett's test (Snedecor & G., 1989) was conducted to check whether the items have equal variances. The test is significant when the correlations of the correlation matrix produced by the PCA differ from null. That is, when the test is not significant, the selected items do not correlate and can therefore not be used for a PCA. For each factor, a scree plot is created which shows the eigenvalues of the extracted factors. The scree test was used as a final check, to see whether only one component with an eigenvalue higher than 1.0 can be extracted based on the selected item set. Finally, the component matrix presents the loading of each item on the extracted factor. A loading is the correlation between the item and the factor.

The PCA was only conducted for the experience variables java programming (exp_jp), refactoring (exp_ref), quality assurance (exp_qs), and software maintenance (exp_main).

C.1.1 Experience in Java Programming (exp_jp)

The resulting disturbing factor exp_jp is composed of the items B4.1, B4.3, B4.4, and B4.5. The resulting statistics are displayed in the following figures and tables.

Table 81 KMO and Bartlett's test for exp_jp

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.790
Bartlett's Test of Sphericity	Approx. Chi-Square	31.218
	df	6
	Sig.	.000

Table 82 Anti-image matrix for exp_jp

	Java APIs (java.util, java.io, java.net, etc.)	Creating Java programs from scratch	Debugging large Java programs	The eclipse IDE (as a user, not a plugin developer)
Java APIs (java.util, java.io, java.net, etc.)	.730(a)	-.341	-.527	-.338
Creating Java programs from scratch	-.341	.860(a)	-.049	-.048
Debugging large Java programs	-.527	-.049	.777(a)	-.264
The eclipse IDE (as user, not a plugin-developer)	-.338	-.048	-.264	.850(a)

a Measures of Sampling Adequacy(MSA)

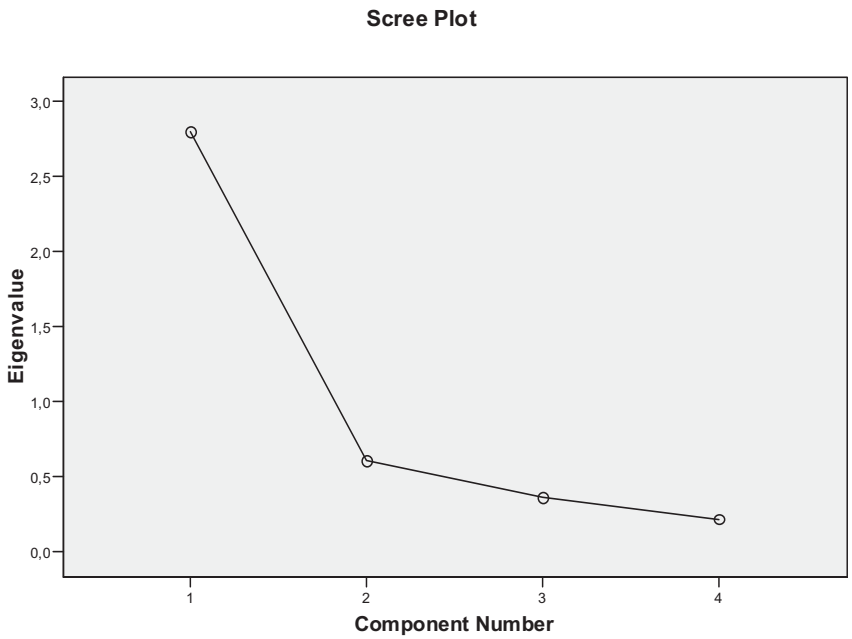


Figure 56 Scree plot for exp_jp

Table 83 Component matrix for exp_java

	Component
	1
Java APIs (java.util, java.io, java.net, etc.)	.913
Creating Java programs from scratch	.721
Debugging large Java programs	.873
The eclipse IDE (as a user, not a plugin developer)	.828

It can be seen that the KMO is 0.79 and that the Bartlett test is significant (p=0.000). In addition, the MSA coefficients are higher than 0.6. Hence, a PCA could be conducted. The sreeplot shows that only one component with an eigenvalue higher than 1.0 could be extracted. The component matrix shows that each item has a high correlation with the extracted factor exp_jp.

C.1.2 Experience in Refactoring (exp_ref)

The resulting disturbing factor exp_ref is composed of the items B5.1, B5.2, B6.2, and B6.3. The resulting statistics are displayed in the following figures and tables.

Table 84 KMO and Bartlett's test for exp_ref

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.687
Bartlett's Test of Sphericity	Approx. Chi-Square	52.072
	df	6
	Sig.	.000

Table 85 Anti-image matrix for exp_ref

	How many years of experience do you have with refactoring?	How many different applications have you refactored? (all programming languages)	Applying Refactorings manually	Applying Refactorings such as "Extract Method" built into an IDE (except the "rename" refactoring)
How many years of experience do you have with refactoring?	.637(a)	.400	.081	-.704
How many different applications have you refactored? (all programming languages)	.400	.681(a)	-.075	-.723
Applying Refactorings manually	.081	-.075	.889(a)	-.400
Applying Refactorings such as "Extract Method" built into an IDE (except the "rename" refactoring)	-.704	-.723	-.400	.619(a)

(a) Measures of Sampling Adequacy(MSA)

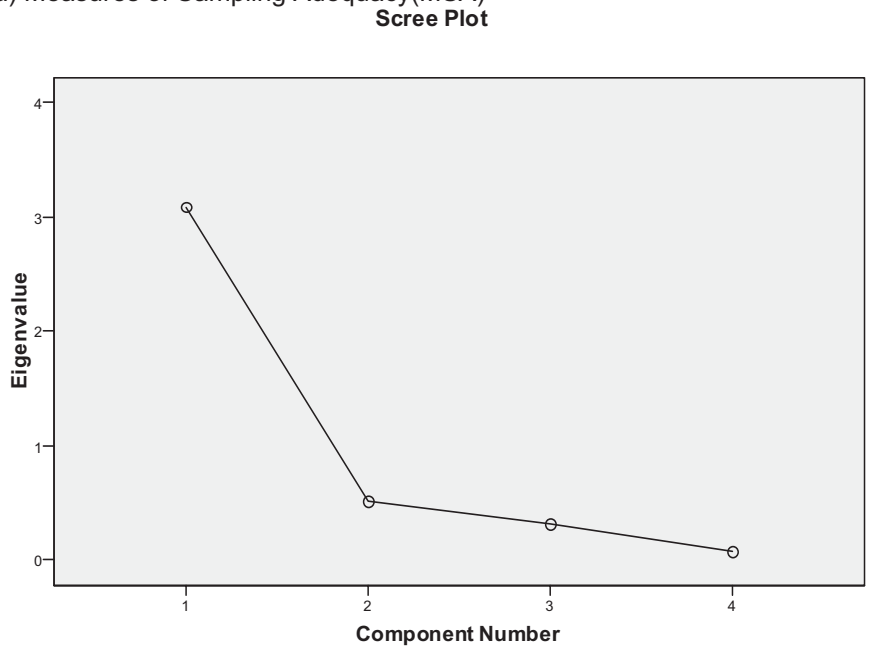


Figure 57 Scree plot for exp_ref

Table 86 Component matrix for exp_ref

	Component
	1
How many years of experience do you have with refactoring?	.800
How many different applications have you refactored? (all programming languages)	.877
Applying Refactorings manually	.861
Applying Refactorings such as "Extract Method" built into an IDE (except the "rename" refactoring)	.970

It can be seen that the KMO is 0.687 and that the Bartlett test is significant (p=0.000). In addition, the MSA coefficients are higher than 0.6 for all items. Hence, a PCA could be conducted. The sreeplot shows that only one component with an eigenvalue higher than 1.0 could be extracted. The component matrix shows that all items have a very high correlation with the extracted factor exp_ref.

C.1.3 Experience in Software Quality Assurance (exp_qs)

The resulting disturbing factor exp_qs is composed of the items B7.1, B7.2, B7.3, and B7.4. The resulting statistics are displayed in the following figures and tables.

Table 87 KMO and Bartlett's test for exp_qs

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.782
Bartlett's Test of Sphericity	Approx. Chi-Square	42.929
	df	6
	Sig.	.000

Table 88 Anti-image matrix for exp_qs

	Quality models (such as ISO 9126.,FURP, Dromey, Boehm, ...)	Testing a software system?	Inspecting a software system regarding quality issues?	Software measurement (Metrics)?
Quality models (such as ISO 9126, FURPS, Dromey, Boehm, …)	.770(a)	-.080	-.064	-.640
Testing a software system?	-.080	.819(a)	-.519	-.124
Inspecting a software system regarding quality issues?	-.064	-.519	.797(a)	-.329
Software measurement (Metrics)?	-.640	-.124	-.329	.751(a)

(a) Measures of Sampling Adequacy(MSA)

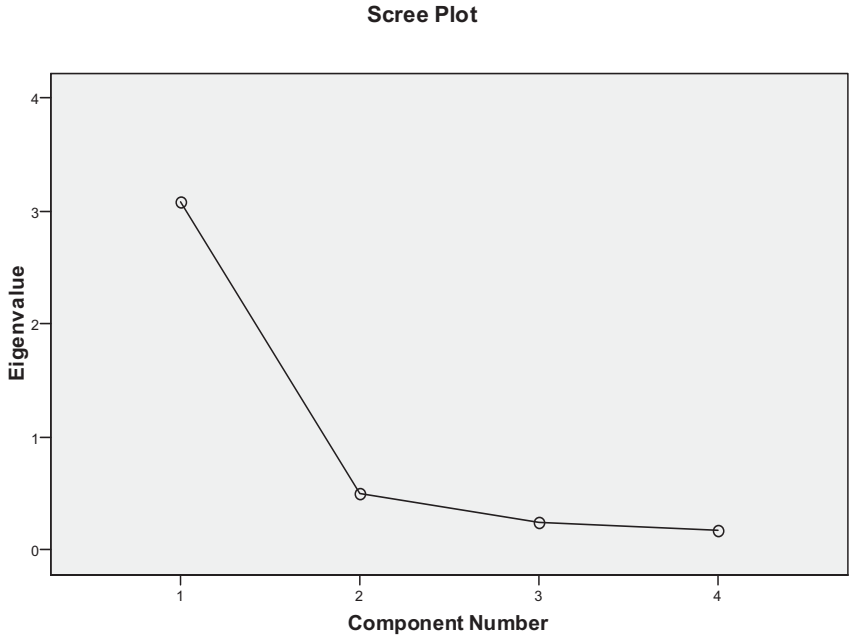


Figure 58 Scree plot for exp_qs

Table 89 Component matrix for exp_qs

	Component 1
Quality models (such as ISO 9126, FURPS, Dromey, Boehm, …)	.865
Testing a software system?	.844
Inspecting a software system regarding quality issues?	.887
Software measurement (Metrics)?	.911

It can be seen that the KMO is 0.782 and that the Bartlett test is significant (p=0.000). In addition, the MSA coefficients are higher than 0.6 for all items. Hence, a PCA could be conducted. The sreeplot shows that only one component with an eigenvalue higher than 1.0 could be extracted. The component matrix shows that all items have a very high correlation with the extracted factor exp_qs.

C.1.4 Experience in Software Maintenance (exp_main)

The resulting disturbing factor exp_dev is composed of the items B8.1, B8.2, B8.3, B8.4, B8.5, and B8.6 The resulting statistics are displayed in the following figures and tables.

Table 90 KMO and Bartlett's test for exp_main

Kaiser-Meyer-Olkin Measure of Sampling Adequacy.		.884
Bartlett's Test of Sphericity	Approx. Chi-Square	108.214
	df	15
	Sig.	.000

Table 91 Anti-image matrix for exp_main

	Maintaining a software system? (e.g., managing defects, applying changes, etc.)	Porting a software system to another platform? (e.g., Java 1.2 to 5.0, Java to C#, etc.)	Improving a software system regarding efficiency (time behavior, resource behavior)?	Improving a software system regarding reliability? (i.e., "Zuverlässigkeit")	Improving a software system regarding usability?	Improving a software system regarding functionality (suitability, interoperability, security)?
Maintaining a software system? (e.g., managing defects, applying changes, etc.)	.934(a)	-.222	.128	-.229	-.257	-.143
Porting a software system to another platform? (e.g., Java 1.2 to 5.0, Java to C#, etc.)	-.222	.885(a)	-.552	-.207	-.128	.096
Improving a software system regarding efficiency (time behavior, resource behavior)?	.128	-.552	.832(a)	-.512	.165	-.132
Improving a software system regarding reliability? (i.e., "Zuverlässigkeit")	-.229	-.207	-.512	.888(a)	-.268	-.030
Improving a software system regarding usability?	-.257	-.128	.165	-.268	.878(a)	-.480
Improving a software system regarding functionality (suitability, interoperability, security)?	-.143	.096	-.132	-.030	-.480	.900(a)

(a) Measures of Sampling Adequacy(MSA)

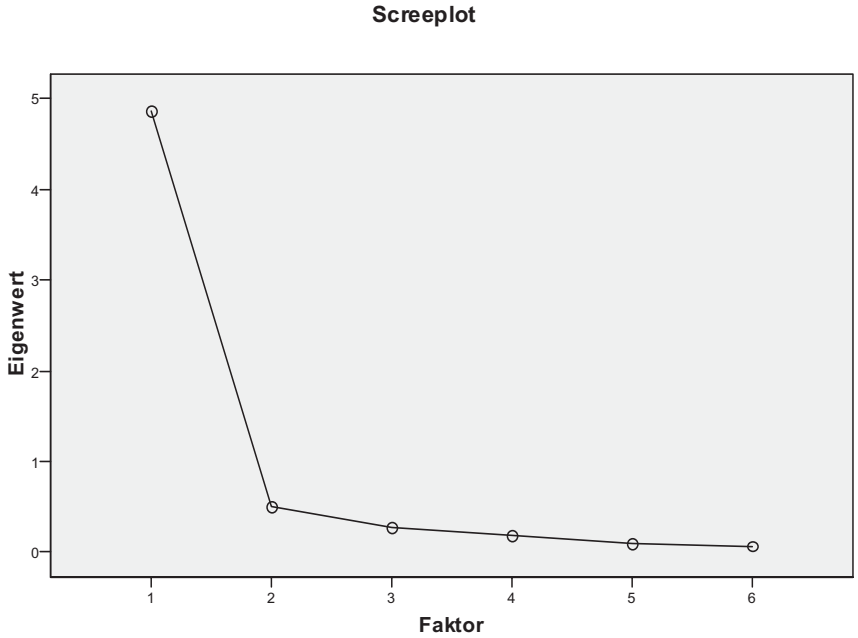


Figure 59 Scree plot for for exp_main

Table 92 Component matrix for exp_main

	Component
	1
Maintaining a software system? (e.g., managing defects, applying changes, etc.)	.887
Porting a software system to another platform? (e.g., Java 1.2 to 5.0, Java to C#, etc.)	.928
Improving a software system regarding efficiency (time behavior, resource behavior)?	.912
Improving a software system regarding reliability? (i.e., "Zuverlässigkeit")	.951
Improving a software system regarding usability?	.894
Improving a software system regarding functionality (suitability, interoperability, security)?	.832

It can be seen that the KMO is 0.884 and that the Bartlett test is significant (p=.00). In addition, the MSA coefficients are higher than 0.6 for all items. Hence, a PCA could be conducted. The sreeplot shows that only one component with an eigenvalue higher than 1.0 could be extracted. The component matrix shows that all items have a very high correlation with the extracted factor exp_main.

C.2. Item Analysis of the Post-Questionnaire

The post-test scores of the second day for all 19 subjects were used for item analysis. As briefly explained in Section 7.3.2, three parameters were calculated for each item of the questionnaire: difficulty index (p), discrimination index (D), and discrimination coefficient (r). These criteria play a role for item selection:

- Negative discrimination indices mean that low scorers perform better than high scorers. These items are probably flawed and should therefore be removed.
- Item difficulty is used to discriminate between students who know the topics and those who do not. Items with an average difficulty around 0 are best for discrimination. However, extreme items with item difficulties close to -1 or 1 should not be deleted by default, since such items are necessary for building comprehensive tests.
- Discrimination coefficient or the point biserial correlation is a correlation between the student's performance on the item (right or wrong) and the total test score. The advantage of this coefficient compared to the discrimination index is that every student who took part in the experiment is used to compute the discrimination coefficient, whereas only 54% (27% upper and 27% lower group) are used to calculate the discrimination indices. Items with a negative discrimination index often have a correlation close to zero or even negative. Those items should be deleted.

Another more implicit criterion was to keep items from each cognitive process category (i.e., remember, understand, analyze, create) that were of interest.

Item analysis is an iterative process where items are deleted one by one. After each deletion, a new reliability analysis is performed. The item analysis values give only a first hint about a necessary deletion. The following table shows the three values for each selected item.

Table 93 Item difficulty, discrimination index, and discrimination coefficient for selected items

Selected item name	Item difficulty (p)	Discrimination index (D)	Discrimination coefficient (r)	Standard deviation
day2_Post1.1	94.7	0.2	0.202	0.23
day2_Post1.2	20.0	0.6	0.256	0.51
day2_Post1.3	100.0	0.0		0.00
day2_Post1.6	89.5	0.2	0.202	0.23
day2_Post2.2	57.9	0.4	0.179	0.42
day2_Post2.3	100.0	0.0		0.00
day2_Post2.5	-47.4	0.2	0.131	0.45
day2_Post2.6	68.4	0.4	0.558	0.37
day2_Post2.7	68.4	0.4	0.311	0.37
day2_Post2.8	-17.6	0.2	0.118	0.50
day2_Post3.1	-14.3	0.4	-0.020	0.48
day2_Post3.10	29.4	0.2	0.191	0.51
day2_Post3.2	100.0	0.0	0.320	0.00
day2_Post3.4	-29.4	0.4	0.211	0.48
day2_Post3.6	-17.6	0.8	0.463	0.50
day2_Post3.7	100.0	0.2	0.298	0.32
day2_Post3.8	25.0	0.4	0.344	0.51
day2_Post4.1	88.9	0.2	0.155	0.32
day2_Post4.2	-78.9	0.4	0.548	0.32
day2_Post4.4	77.8	0.0	0.036	0.37
day2_Post4.6	88.9	0.0	-0.007	0.32
day2_Post5.10	-100.0	0.0	0.202	0.23
day2_Post5.13	47.4	0.2	0.104	0.68
day2_Post5.14	60.0	0.6	0.282	0.70
day2_Post5.15	-33.3	0.4	0.211	0.78
day2_Post5.16	53.8	0.2	0.203	0.69
day2_Post5.6	71.4	0.2	0.202	0.23
day2_Post5.7	100.0	0.2	0.513	0.23
day2_Post5.8	-66.7	0.2	0.081	0.77
day2_Post5.9	-100.0	0.0	0.298	0.32
PostC2.2	-15.8	0.4	0.213	1.13
PostC2.3	-15.8	0.4	0.213	1.13

Selected item name	Item difficulty (p)	Discrimination index (D)	Discrimination coefficient (r)	Standard deviation
PostC2.4	-100.0	0.0	0.038	1.36
PostC2.5	-85.7	0.0	0.206	0.73
PostC2.6.1	33.3	0.0	-0.012	0.50
PostC2.6.2	22.2	0.2	0.102	0.51
PostC3.1	68.4	0.0	0.070	0.37
PostC3.2.1	33.3	0.4	-0.035	0.51
PostC3.2.2	88.2	0.4	0.381	0.37
PostC3.2.3	12.5	0.2	0.017	0.51
PostC3.2.4	75.0	0.2	0.062	0.45
PostC3.3	44.4	0.4	0.259	0.48
PostC3.5	88.2	0.4	0.329	0.37
PostC3.6	86.7	0.4	0.291	0.45
PostC3.7.1	57.1	0.4	0.346	0.51
PostC3.7.2	42.9	0.2	0.216	0.51
PostC4.1	78.9	0.2	0.257	0.32
PostC4.2.2	17.6	0.4	0.318	0.51
PostC4.2.3	22.2	0.2	0.026	0.51
PostC4.2.5	36.8	0.6	0.580	0.48
PostC4.3	33.3	0.4	0.435	0.50
PostC4.4	100.0	0.0		0.00
PostC4.6	-17.6	0.4	0.302	0.50
PostC4.7.1	-100.0	0.0	0.327	1.02
PostC4.7.2	-100.0	0.0	0.351	0.99
PostC5.2.3	57.1	0.6	0.140	0.51
PostC5.2.4	68.4	0.4	0.434	0.37
PostC5.2.5	15.8	0.2	0.001	0.51
PostC5.2.6	89.5	0.0	0.035	0.23
PostC5.3	33.3	0.2	0.078	0.50
PostC5.4	100.0	0.2	0.401	0.32
PostC5.5	78.9	0.2	0.175	0.32
PostC5.6	50.0	0.6	0.395	0.50
PostC5.7.1	29.4	0.2	0.392	0.83
PostC5.7.2	25.0	0.4	0.369	0.92

The next figure shows a scatter plot for item difficulty and discrimination coefficient. The plot helps to easily identify items with a negative discrimination coefficient and extreme item difficulties, i.e., close to -1 or 1.

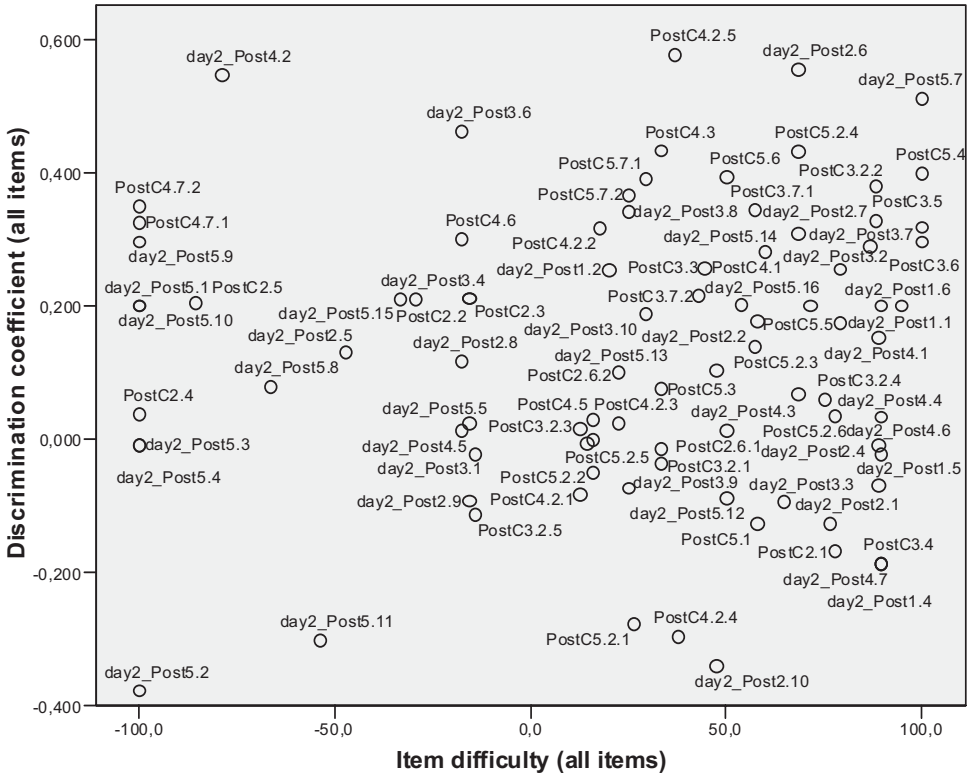


Figure 60 Scatter plot for discrimination index and item difficulty

The next table shows the item analysis values for the deleted items, i.e., those items that were not used for the calculation of the knowledge acquisition variable.

Table 94 Item difficulty, discrimination index, and discrimination coefficient for deleted items

Deleted item name	Item difficulty (p)	Discrimination index (D)	Discrimination coefficient (r)	Standard deviation
day2_Post1.4	89.5	0.0	-0.186	0.23
day2_Post1.5	88.9	0.0	-0.068	0.32
day2_Post2.1	76.5	0.0	-0.125	0.42
day2_Post2.10	47.4	-0.4	-0.340	0.45
day2_Post2.4	89.5	0.0	-0.020	0.23
day2_Post2.9	-15.8	-0.2	-0.091	0.51
day2_Post3.3	64.7	-0.4	-0.093	0.45
day2_Post3.5	14.3	0.0	-0.004	0.51
day2_Post3.9	25.0	-0.2	-0.072	0.51
day2_Post4.3	50.0	-0.2	0.014	0.50
day2_Post4.5	-17.6	0.2	0.014	0.50
day2_Post4.7	89.5	0.0	-0.186	0.23
day2_Post5.1	-100.0	0.0	0.202	0.23
day2_Post5.11	-53.8	0.0	-0.301	0.61
day2_Post5.12	50.0	0.2	-0.088	0.32
day2_Post5.2	-100.0	0.0	-0.376	0.50

Deleted item name	Item difficulty (p)	Discrimination index (D)	Discrimination coefficient (r)	Standard deviation
day2_Post5.4	-100.0	0.0	-0.007	0.56
day2_Post5.5	-15.8	0.4	0.026	0.51
PostC2.1	77.8	-0.2	-0.167	0.37
PostC3.2.5	-14.3	-0.2	-0.111	0.48
PostC3.4	89.5	0.0	-0.186	0.23
PostC4.2.1	12.5	0.0	-0.082	0.51
PostC4.2.4	37.5	-0.2	-0.295	0.51
PostC4.5	15.8	0.4	0.031	1.03
PostC5.1	57.9	0.0	-0.125	0.42
PostC5.2.1	26.3	-0.2	-0.277	0.50
PostC5.2.2	15.8	-0.2	-0.049	0.51

It seems that most of the deleted items possess extreme item difficulties (those close to -100 are very difficult and those close to +100 are very easy), negative or zero discrimination indices, and negative or very low discrimination coefficients. The next figure shows the scatterplot of the remaining items.

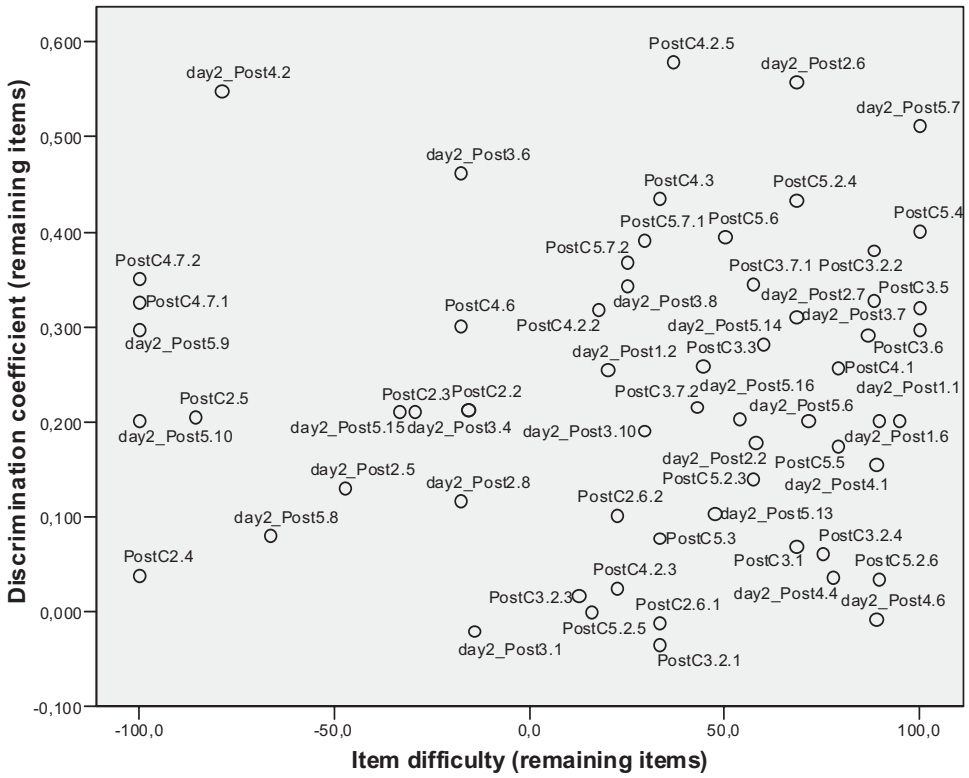


Figure 61 Scatter plot for discrimination coefficient and item difficulty of remaining items

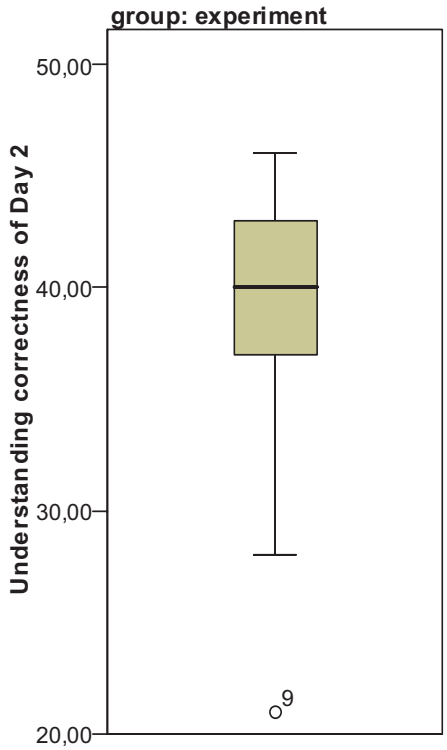
The plot shows that almost two thirds of the items tend to be easier, which means that the students were able to answer most of the questions correctly after two days of experimentation. Nevertheless, some items were solved only by a few students or were answered wrongly

many times (which leads to negative item difficulties). There are still a few items with a negative discrimination coefficient. By checking these items based on the scores of the first day, it seems that these items are still suitable for the questionnaire and necessary to balance the questionnaire in terms of cognitive process dimensions.

C.3. Outlier Analysis

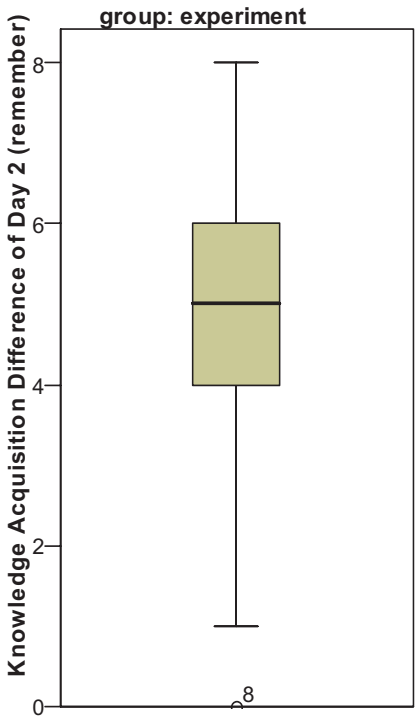
The following outliers were detected:

Disturbing factor experience refactoring (exp_ref): One outlier (z-value of 2.18) was detected (value=4.75; mean=2.07), which corresponds to a very low level of experience in refactoring. However, this value was kept because the subject also had lower ratings for the other experience levels.



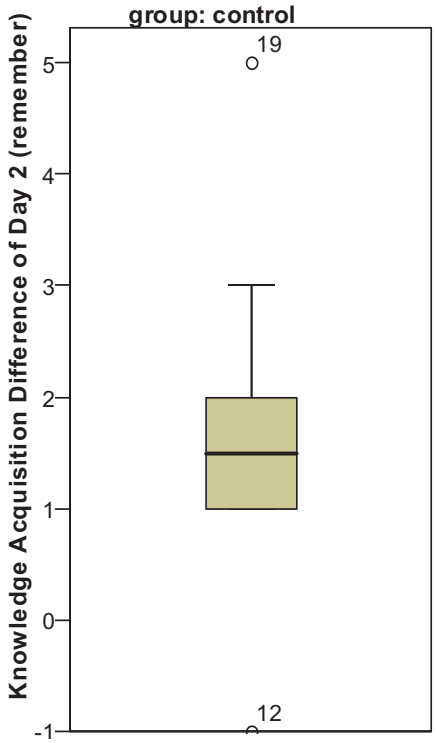
ucorr (experimental group, day2): One outlier was identified (z-value=-2.01, mean=37.88, value=21), which was very low understanding correctness compared to the other subjects in the group. The reason was not a lack of time for answering the questions, because the subject did not choose this in the debriefing questionnaire. Therefore, the deviation could not be explained. It was decided to omit the data from further analysis. The following figure shows the boxplot

Figure 62 Box-and-whisker plot for ucorr (experimental group, day 2)



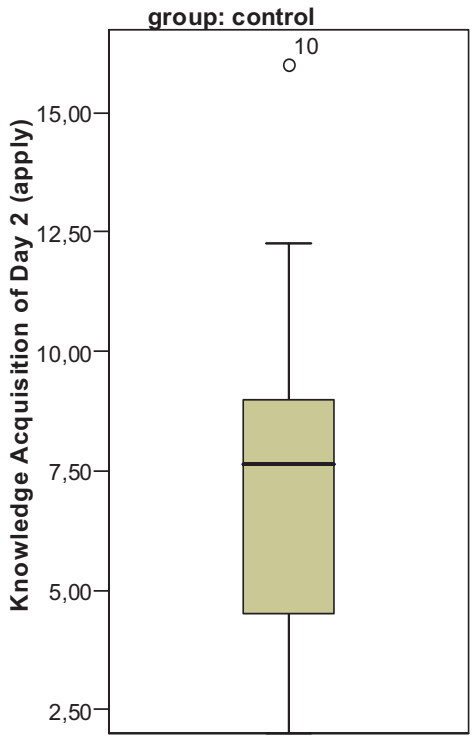
know_diff_remember (experimental group, day2): One outlier was identified (z-value=1.71, mean=4.56, value=0), which was a rather low score difference compared to the other subjects in the group. It was decided to omit the data from further analysis. The following figure shows the boxplot

Figure 63 Box-and-whisker plot for know_diff_remember (experimental group, day 2)



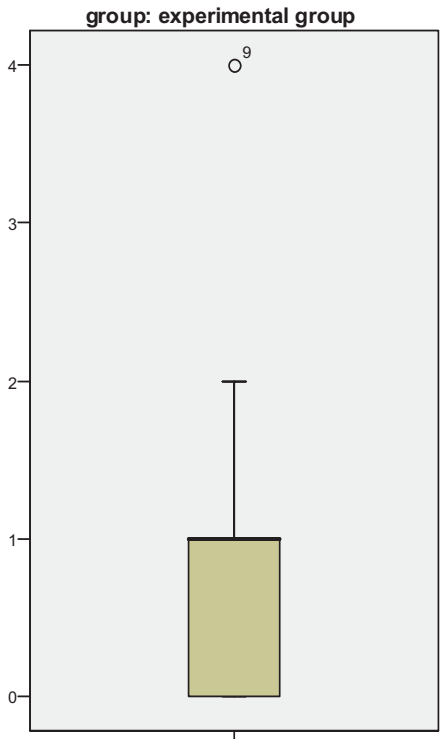
know_diff_remember (control group, day2): Two outliers were identified (1: z-value=-1.72, mean=1.70, value=-1; 2: z-value=2.10, mean=1.70, value=5), which was a low, respectively high, score difference compared to the other subjects in the group. It was decided to omit the data from further analysis. The following figure shows the boxplot

Figure 64 Box-and-whisker plot for know_diff_remember (control group, day 2)



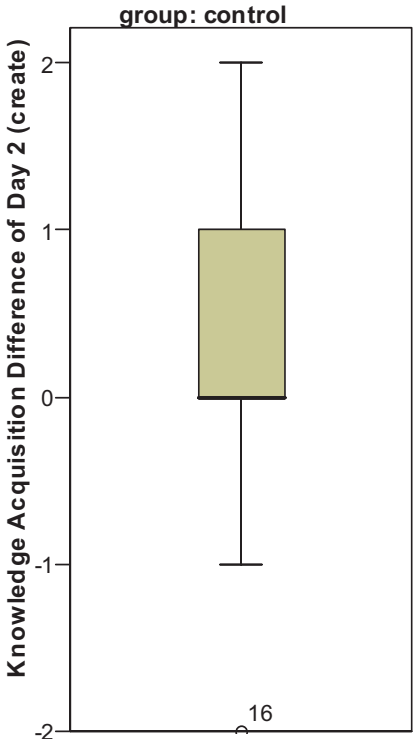
know_diff_apply (control group, day2): One outlier was identified (z-value=, mean=1.70, value=16), which was a very high score difference compared to the other subjects in the group. It was decided to omit the data from further analysis. The following figure shows the boxplot

Figure 65 Box-and-whisker plot for know_diff_remember (control group, day 2)



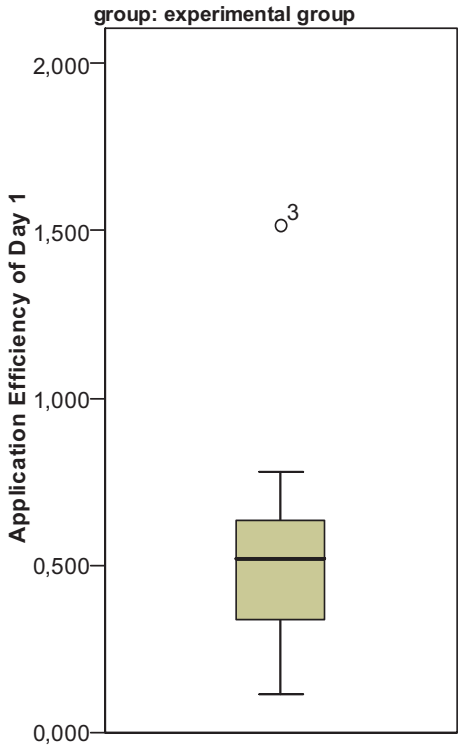
know_diff_create (experimental group, day1): One outlier was identified (z-value=2.42, mean=1.10, value=4), which is an extreme outlier. The values were checked again and were correct. The subject performed very well in this category of questions. The values were changed or omitted for further analysis because this subject also got the highest score during the second period in the control group.

Figure 66 Box-and-whisker plot for know_diff_create (control group, day 1)



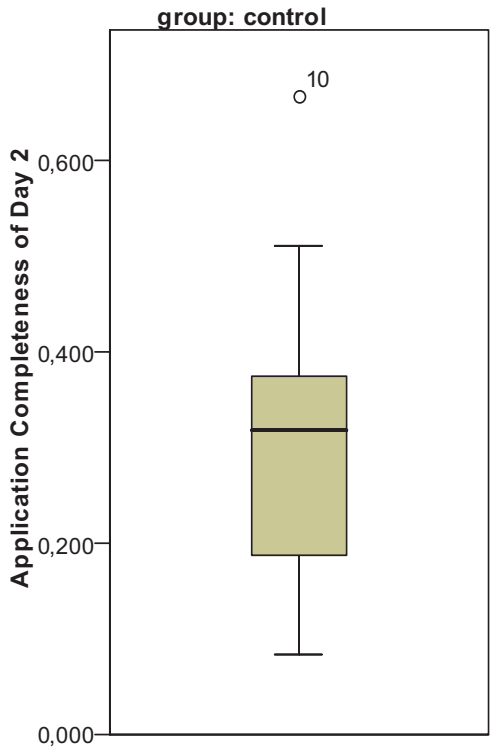
know_diff_create (control group, day2): One outlier was identified (z-value=-193, mean=0.20, value=-2), which was a very low score difference compared to the other subjects in the group. It was decided to omit the data from further analysis. The following figure shows the boxplot

Figure 67 Box-and-whisker plot for know_diff_create (control group, day 2)



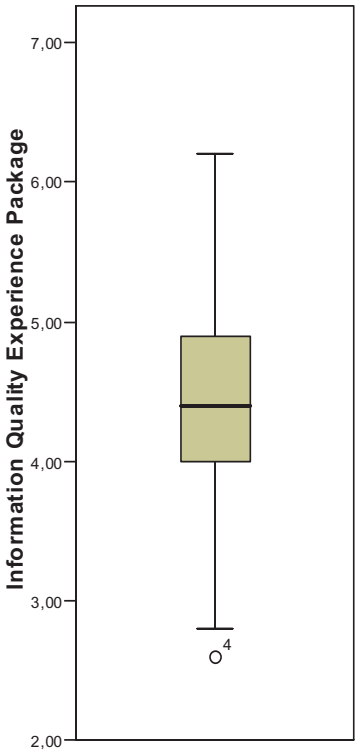
aeff (experimental group, day1): One outlier was identified (z-value=2.42, mean=0.55, value=1.52), which was very high efficiency compared to the other subjects in the group. The reason for this high value was the very high efficiency of the second assignment where the subject performed all refactorings correctly within a very short period of time (i.e., 5 minutes) compared to the other subjects. It was decided to omit the data from further analysis. The following figure shows the boxplot

Figure 68 Box-and-whisker plot for aeff (experimental group, day 1)



acomp (control group, day2): One outlier was identified (z-value=1.94, mean=0.32, value=0.67), which was a very high score difference compared to the other subjects in the group. It was decided to omit the data from further analysis. The following figure shows the boxplot

Figure 69 Box-and-whisker plot for acomp (control group, day 2)



inf_qua (both groups, experience package, debriefing questionnaire): One outlier was identified (z-value=-1.728, mean=4.42, value=2.60) which was a very low score difference compared to the other subjects in the group. However, since the z-value is still acceptable and the outlier was not an extreme one, it was decided not to omit the value. The following figure shows the boxplot

Figure 70 Box-and-whisker plot for (both groups, experience package, debriefing questionnaire)

No outliers were identified for the other data sets. All further analysis steps were done without the previously identified outliers.

C.4. Test for Normality

The simplest method for assessing normality is to look at the frequency distribution histogram, boxplot, or stem-and-leaf plot. The most important things to look at are the symmetry and peakiness of the curves. In addition, curves that indicate two or more peaks would show a bimodal distribution and are not suitable for parametric statistics. Frequency distribution histograms must only be used as an indication of the distribution, and subsequently, better methods must be used. Values of skewness and kurtosis are good indicators, but can be overly optimistic regarding the data's match with normality. Graphical methods are intuitive and easy to interpret, while numerical methods provide more objective ways of examining normality. Therefore, numerical methods for normality (and homogeneity of variance) should always be carried out as a best practice in statistics. In this work, the SPSS Shapiro-Wilk test (Shapiro & Wilk, 1965) was used, which is more reliable when $n<50$ is applied to all measures. The outcome of the Kolmogorov-Smirnov test, which is the principal goodness-of-fit test for normal and uniform data sets should be used with care, since this test is not very suitable for small samples, in the case of this experiment.

Both of the above tests use the same hypotheses:

> H_0: There is no difference between the distribution of the data set and a normal one.
> H_1: There is a difference between the distribution of the data set and a normal one.

In addition, theory driven plots (i.e., Q-Q plots) were used for testing normality in those cases where H_0 could not be rejected at a significance level lower than 0.05. The *quantile-quantile plot* (Q-Q plot) compares ordered values of a variable with quantiles of a specific theoretical distribution (i.e., the normal distribution). If two distributions match, the points on the plot will form a linear pattern passing through the origin with a unit slope. Detrended normal Q-Q plots depict the actual deviations of data points from the straight horizontal line at zero. No specific pattern in a detrended plot indicates normality of the variable.

Deviations from the normality distribution function can be easily detected since detrended normal Q-Q and Q-Q plots depict the actual deviations of data points from the straight horizontal line at zero.

C.4.1 Disturbing Factors

The following table shows the outcome of the Shapiro Wilk test and the Kolmogorov-Smirnov test.

Table 95 Test for normality for experience level variables

	Disturbing Factors	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Software Development Experience	exp_dev	.172	19	.172	.932	19	.191
Java Experience	exp_jp	.164	19	.195	.948	19	.360
Refactoring experience	exp_ref	.210	19	.027	.819	19	***.002***
SQA experience	exp_sqa	.170.	19	0.150	.933	19	.195
Software maintenance experience	exp_main	.150	19	.200	.891	19	***.034***
Time need	tn	.326	19	.000	.768	19	***.000***
Information quality of learning space	inf_qua (LSEP)	.132	19	.200(*)	.982	19	.963
Information quality of experience package	inf_qua (EP)	.134	19	.200(*)	.955	19	.480
Pre-test score	pre-test	,118	19	,200(*)	,946	19	,339

* This is a lower bound of the true significance.

It can be seen that the disturbing factors exp_ref and exp_main have a higher significance level of 0.05, which means that it can be assumed that these factors are not normally distributed.

The following figures show the histogram, boxplot, and Q-Q-plot for the disturbing factors.

C.4.1.1 Experience in Refactoring (exp_ref).

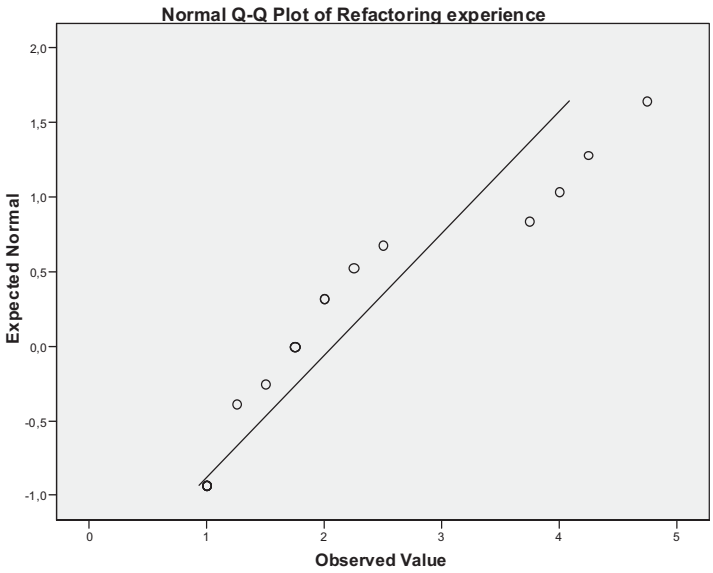


Figure 71 Q-Q-Plot for refactoring experience (exp_ref)

Figure 72 Detrended Q-Q-Plot for refactoring experience (exp_ref)

The Q-Q plot shows for every data set a high deviation from normality. In addition, the detrended normal Q-Q plot also shows higher deviations from normal and a pattern could be detected in the lower range of the observed value. Therefore, exp_ref cannot be considered as a normally distributed variable. The reason for this is depicted in the histogram and the boxplot shows on the one side a high skewness of 1.11 (a situation's asymmetry in relation to a normal distribution) and that most of the subjects had almost no experience in refactoring.

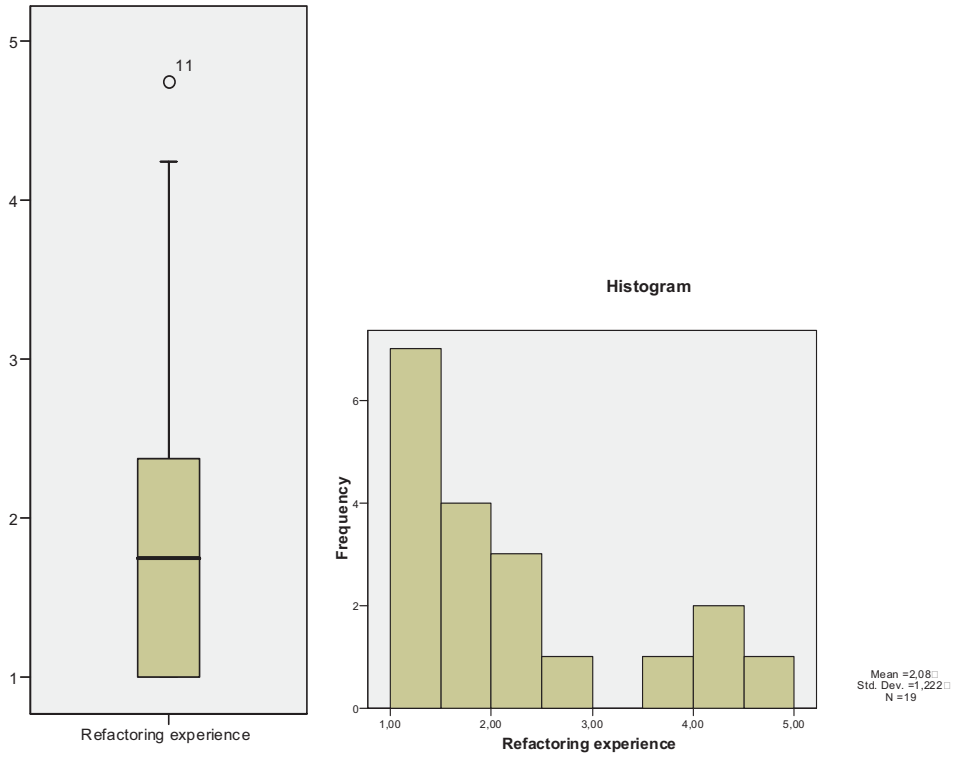


Figure 73 Histogram and boxplot for refactoring experience (exp_ref)

C.4.1.2 Experience in Maintenance (exp_main).

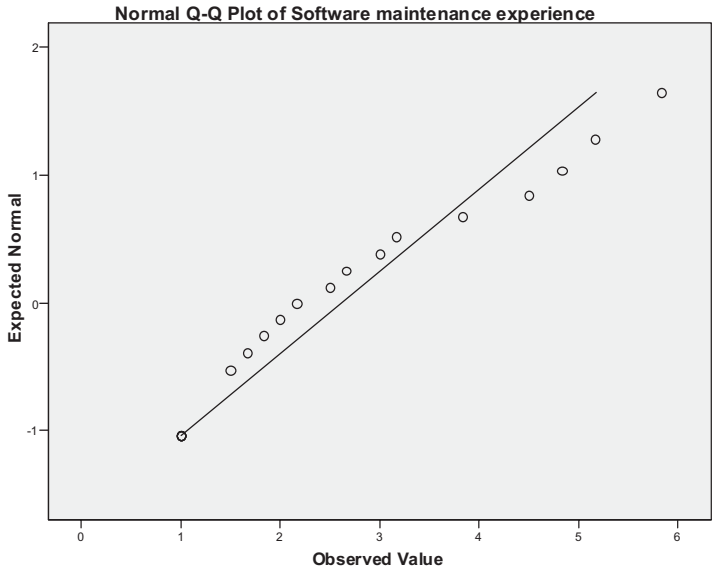


Figure 74 Q-Q plot for refactoring experience (exp_main)

The Q-Q plot shows for all data sets a deviation from normality. In addition, the detrended normal Q-Q plot also shows deviations from a normal distribution and two data clusters/patterns could be detected in the lower and higher range of the observed values. Therefore, exp_ref cannot be considered as a normally distributed variable. The reason for this is depicted in the histogram and the boxplot shows on the one side a high skewness of 0.75 and that most of the subjects had almost no experience in software maintenance. Hence, for the disturbing factors exp_ref and exp_main, parametric methods cannot be used or should be used only carefully and only when a non-parametric method is used in addition for checking the outcome of the parametric one.

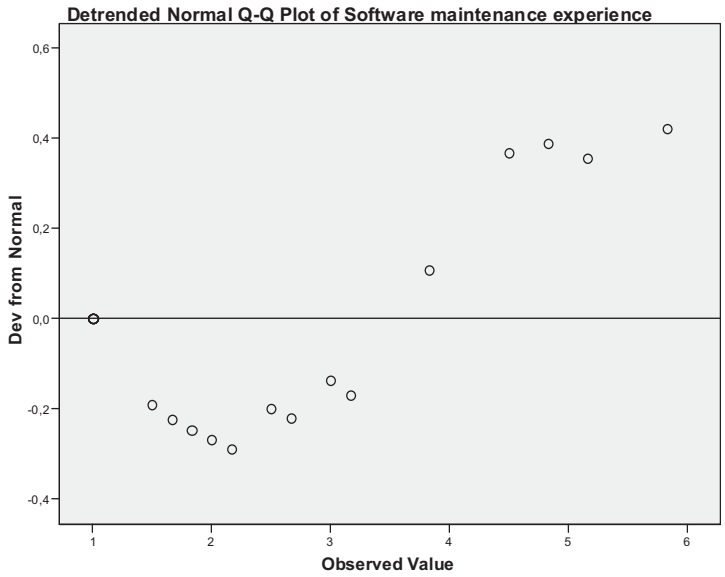


Figure 75 Detrended Q-Q plot for refactoring experience (exp_main)

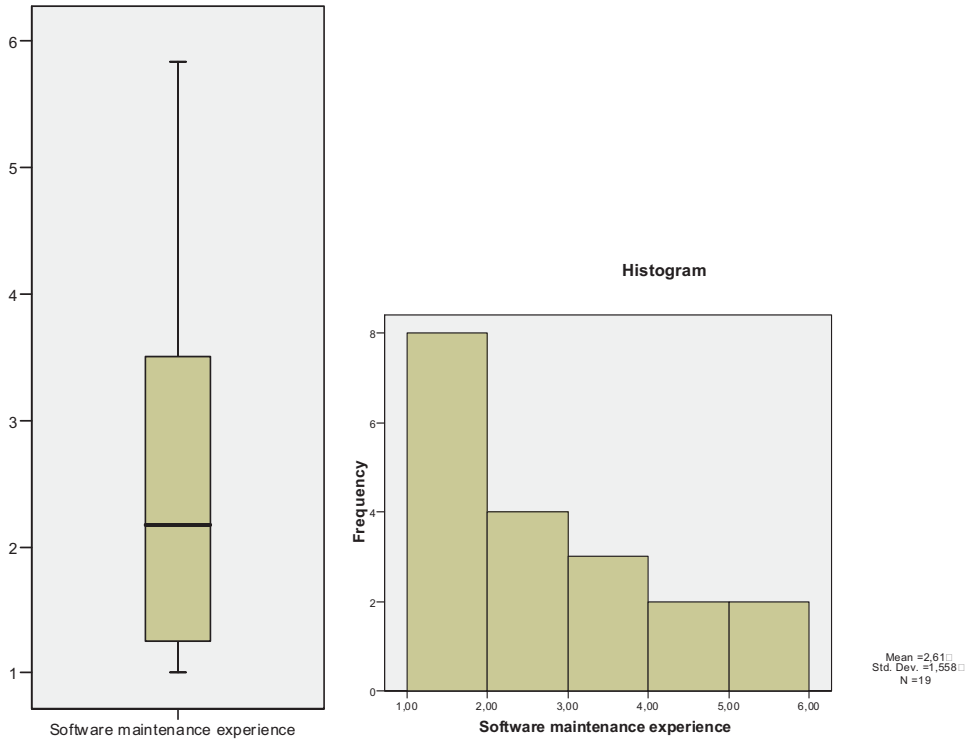


Figure 76 Histogram and boxplot for refactoring experience (exp_main)

C.4.1.3 Time Need (tn)

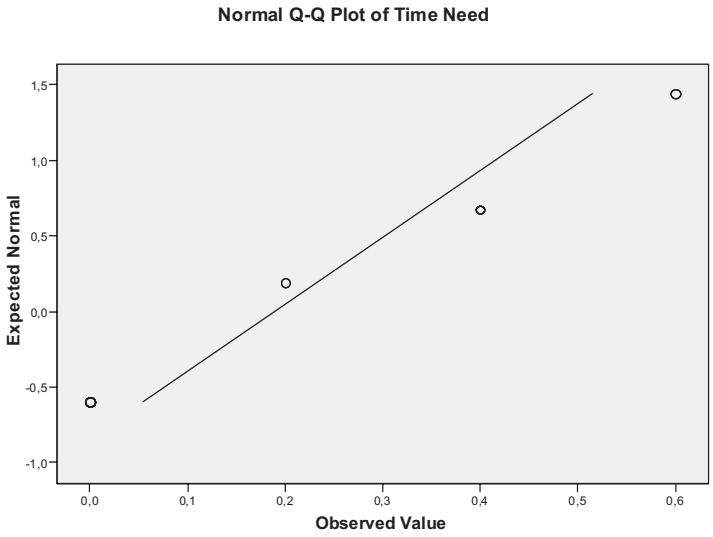


Figure 77 Q-Q plot for time need (tn)

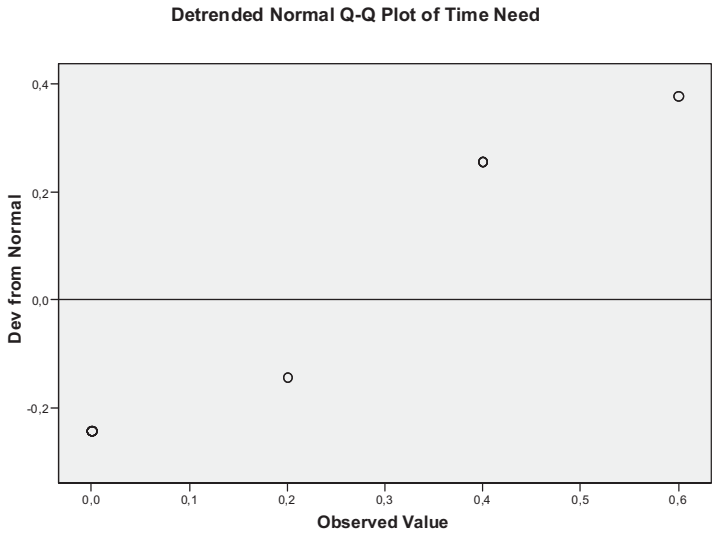


Figure 78 Detrended Q-Q plot for time need (tn)

Only four different values were available for the control group during the first day, which makes it impossible to detect a pattern/data cluster in the detrended Q-Q plot. The Q-Q plot shows a strong deviation from normality for two values. In fact, know_diff_create cannot be considered as a normally distributed variable. The reason for this is depicted in the histogram and boxplot. The histogram shows on the one side a medium skewness of 0.63 (a situation's asymmetry in relation to a normal distribution) and that most of the subjects had no lack of time for reading the information, solving the assignments, and getting familiar with the Wiki and the learning space.

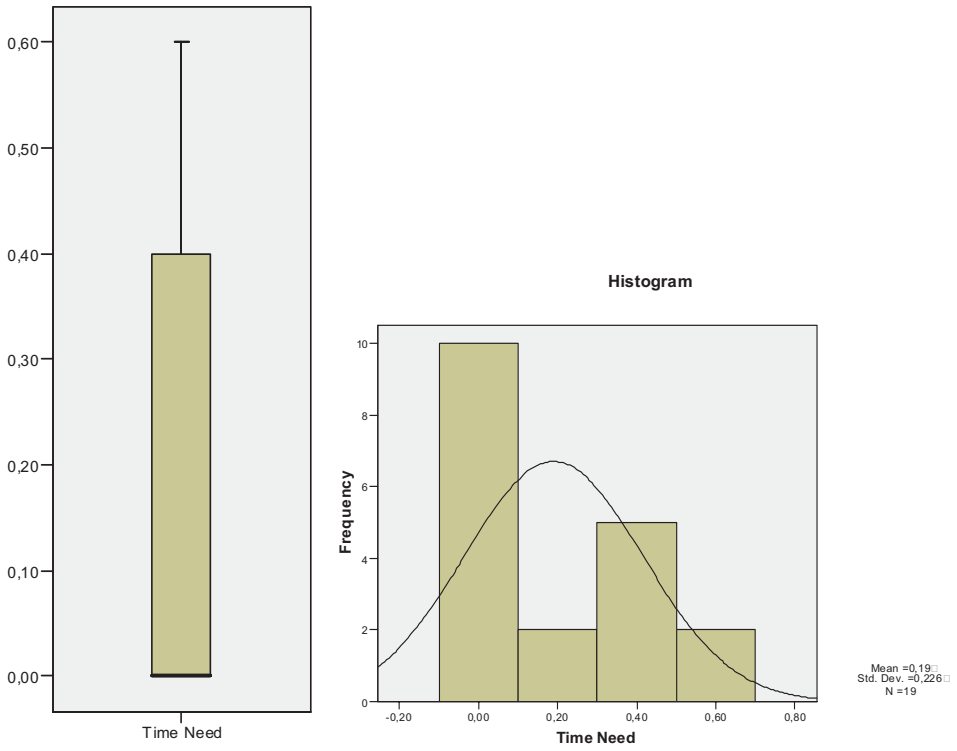


Figure 79 Histogram and boxplot time need (tn)

C.4.2 Dependent Variables (for both groups and both periods)

The following table shows the outcome of the Shapiro-Wilk test and the Kolmogorov-Smirnov test for the experimental and control groups for both days separately. The outliers were removed from the data sets before the normality test was performed.

Table 96 Test for normality for dependent variables (experimental group)

	Dependent Variables	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Understanding correctness of Day 1	ucorr	.113	10	.200(*)	.962	10	.803
Understanding correctness of Day 2	ucorr	.180	8	.200(*)	.896	8	.267
Knowledge Acquisition Difference of Day 1	know_diff	.143	10	.200(*)	.937	10	.519
Knowledge Acquisition Difference of Day 2	know_diff	.223	9	.200(*)	.861	9	.097
Knowledge Acquisition Difference of Day 1 (remember)	know_diff_remember	.209	10	.200(*)	.899	10	.213
	Dependent Variables	**Kolmogorov-Smirnov(a)**			**Shapiro-Wilk**		
		Statistic	**df**	**Sig.**	**Statistic**	**df**	**Sig.**
Knowledge Acquisition Difference of Day 2 (remember)	know_diff_remember	.177	8	.200(*)	.951	8	.720
Knowledge Acquisition Difference of Day 1 (understand)	know_diff_understand	.242	10	.102	.928	10	.429
Knowledge Acquisition Difference of Day 2 (understand)	know_diff_understand	.186	9	.200(*)	.933	9	.514
Knowledge Acquisition of Day 1 (apply)	know_diff_apply	.155	10	.200(*)	.943	10	.587
Knowledge Acquisition of Day 2 (apply)	know_diff_apply	.216	9	.200(*)	.886	9	.183
Knowledge Acquisition Difference of Day 1 (analyze)	know_diff_analyze	.280	10	.025	.805	10	***.017***
Knowledge Acquisition Difference of Day 2 (analyze)	know_diff_analyze	.284	9	.035	.908	9	.303
Knowledge Acquisition Difference of Day 1 (create)	know_diff_create	.333	10	.002	.778	10	***.008***
Knowledge Acquisition Difference of Day 2 (create)	know_diff_create	.269	9	.059	.808	9	***.025***
Application Efficiency of Day 1 (based on score of marks and refactoring)	aeff	.177	9	.200(*)	.960	9	.803
Application Efficiency of Day 2 (based on score of marks and refactoring)	aeff	.134	9	.200(*)	.936	9	.537
Application Completeness of Day 1 (based on score of marks and refactoring)	acomp	.155	10	.200(*)	.943	10	.587
Application Completeness of Day 2 (based on score of marks and refactoring)	acomp	.216	9	.200(*)	.886	9	.183
Application Accuracy of Day 1	aaccu	.137	10	.200(*)	.976	10	.943

	Dependent Variables	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Application Accuracy of Day 2	aaccu	.145	9	.200(*)	.945	9	.638
Information quality of learning space	inf_qua (LSEP)	,132	19	,200(*)	,982	19	,963
Information quality of experience package	inf_qua (EP)	,134	19	,200(*)	,955	19	,480

* This is a lower bound of the true significance. a Lilliefors Significance Correction

Table 97 Test for normality for dependent variables (control group)

	Dependent Variables	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
		Statistic	df	Sig.	Sta-tistic	df	Sig.
Understanding correctness of Day 1	ucorr	.204	9	.200(*)	.932	9	.497
Understanding correctness of Day 2	ucorr	.185	10	.200(*)	.913	10	.299
Knowledge Acquisition Difference of Day 1	know_diff	.171	9	.200(*)	.896	9	.228
Knowledge Acquisition Difference of Day 2	know_diff	.222	10	.178	.905	10	.249
Knowledge Acquisition Difference of Day 1 (remember)	know_diff_remember	.259	9	.083	.844	9	.065
Knowledge Acquisition Difference of Day 2 (remember)	know_diff_remember	.300	8	.033	.798	8	***.027***
Knowledge Acquisition Difference of Day 1 (understand)	know_diff_understand	.176	9	.200(*)	.927	9	.452
Knowledge Acquisition Difference of Day 2 (understand)	know_diff_understand	.160	10	.200(*)	.973	10	.914
Knowledge Acquisition of Day 1 (apply)	know_diff_apply	.171	9	.200(*)	.952	9	.714
	Dependent Variables	**Kolmogorov-Smirnov(a)**			**Shapiro-Wilk**		
		Statistic	**df**	**Sig.**	**Sta-tistic**	**df**	**Sig.**
Knowledge Acquisition of Day 2 (apply)	know_diff_apply	.170	9	.200(*)	.962	9	.822
Knowledge Acquisition Difference of Day 1 (analyze)	know_diff_analyze	.160	9	.200(*)	.945	9	.639
Knowledge Acquisition Difference of Day 2 (analyze)	know_diff_analyze	.297	10	.013	.868	10	.095
Knowledge Acquisition Difference of Day 1 (create)	know_diff_create	.272	9	.054	.805	9	***.024***
Knowledge Acquisition Difference of Day 2 (create)	know_diff_create	.248	9	.116	.913	9	.338
Application Efficiency of Day 1 (based on score of marks and refac-toring)	aeff	.143	9	.200(*)	.946	9	.646
Application Efficiency of Day 2 (based on score of marks and refac-toring)	aeff	.186	10	.200(*)	.929	10	.434
Application Completeness of Day 1 (based on score of marks and refac-toring)	acomp	.171	9	.200(*)	.952	9	.714
Application Completeness of Day 2 (based on score of marks and refac-toring)	acomp	.170	9	.200(*)	.962	9	.822

Application Accuracy of Day 1	aaccu	.137	9	.200(*)	.983	9	.980
Application Accuracy of Day 2	aaccu	.162	10	.200(*)	.969	10	.879

* This is a lower bound of the true significance. a Lilliefors Significance Correction

It can be seen that for several dependent variables (in bold-italic type), the p-value of the Shapiro-Wilk test is not higher than 0.05, which means that it can be assumed that these factors are not normally distributed. The following figures show the histograms, boxplots, and (detrended) Q-Q plots for these dependent variables for further analysis of the deviations from a normal distribution.

Table 98 Test for normality for dependent variables informatin quality

	Dependent Variables	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Information quality of learning space	inf_qua (LSEP)	.132	19	.200(*)	.982	19	.963
Information quality of experience package	inf_qua (EP)	.134	19	.200(*)	.955	19	.480

* This is a lower bound of the true significance. a Lilliefors Significance Correction

C.4.2.1 Knowledge Acquisition Difference Remember Control Day 2 (know_diff_remember)

Only three different values where available for the control group during the first day, which makes it impossible to detect for example a pattern/data cluster in the detrended Q-Q-Plot. The Q-Q plot shows a strong deviation from normality for two values. In fact, know_diff_remember cannot be considered as a normal distributed variable for the data set control group/Day 2. The reason for this is depicted in the histogram and boxplot, which show on the one side a skewness of 0.824 and that most of the subjects perform bad for the cognitive process level *remember*.

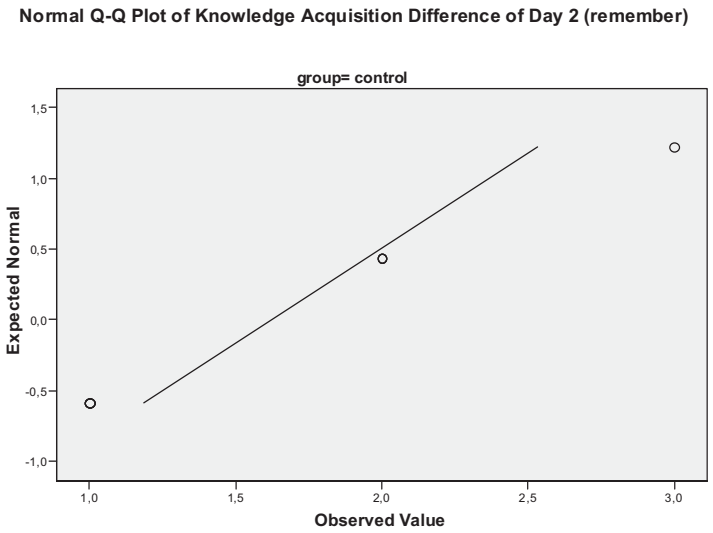


Figure 80 Q-Q plot for knowledge acquisition difference remember day 2 control group (know_diff_remember)

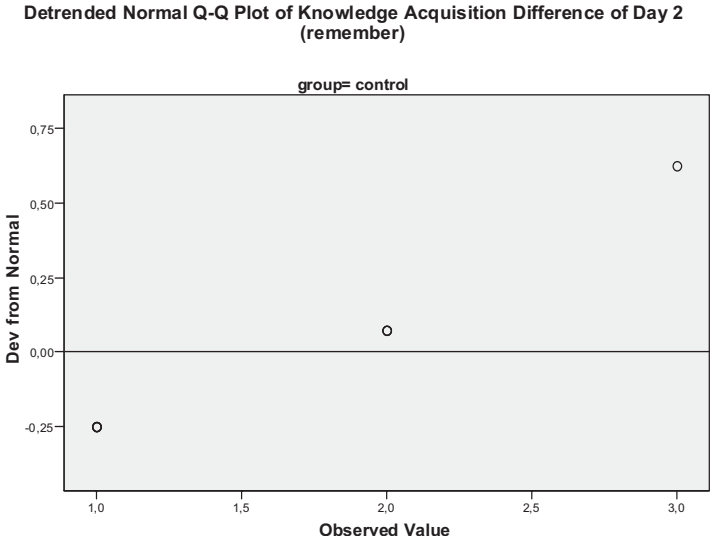


Figure 81 Detrended Q-Q plot for knowledge acquisition difference remember day 2 control (know_diff_ remember)

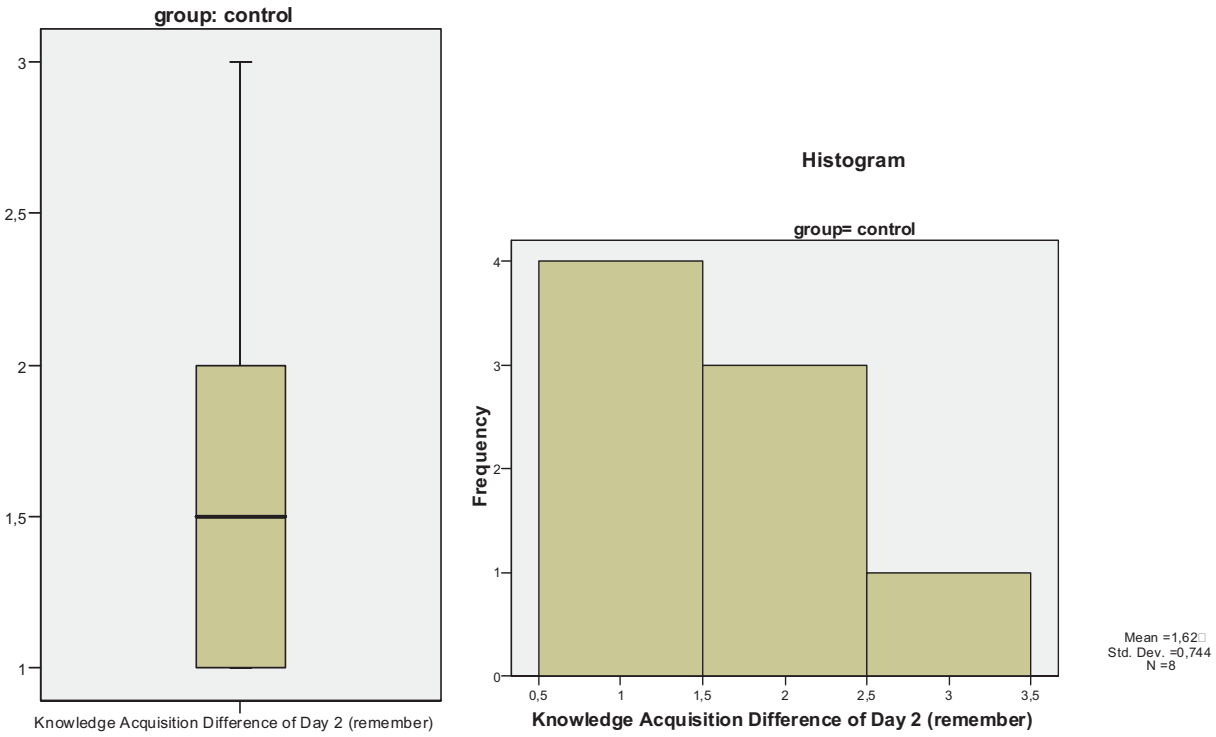


Figure 82 Histogram and boxplot for knowledge acquisition difference remember day 2 control (know_diff_ remember)

C.4.2.2 Knowledge Acquisition Difference Create Experimental Day 1 (know_diff_create)

The Q-Q plot shows for the data set a deviation from normality, at least for the value 4. As for the know_diff_analyze variable no cluster/pattern can be detected in the detrended normal Q-Q plot. Nevertheless, the Q-Q plot confirms that the deviations are to large to consider this data set as normally distributed. The reason for this is depicted in the histogram and the boxplot, which shows on the one side a very high skewness of

1.709 and that most of the subjects almost performed badly on average for this cognitive process level.

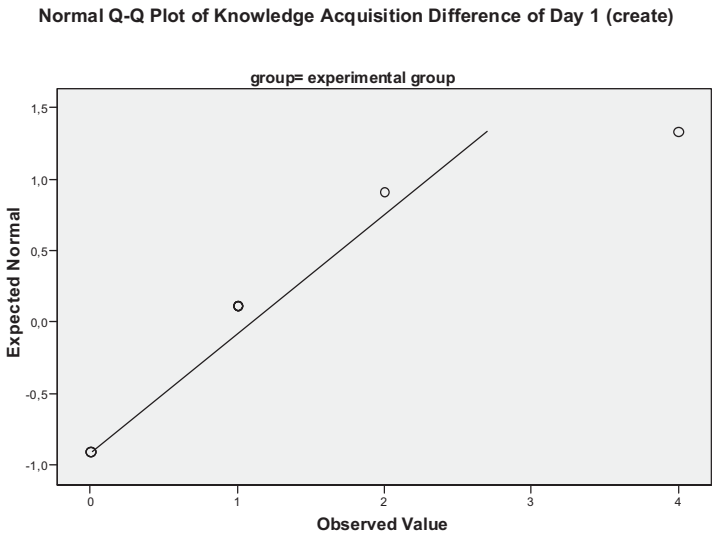


Figure 83 Q-Q plot for knowledge acquisition difference create day 1 experimental group (know_diff_create)

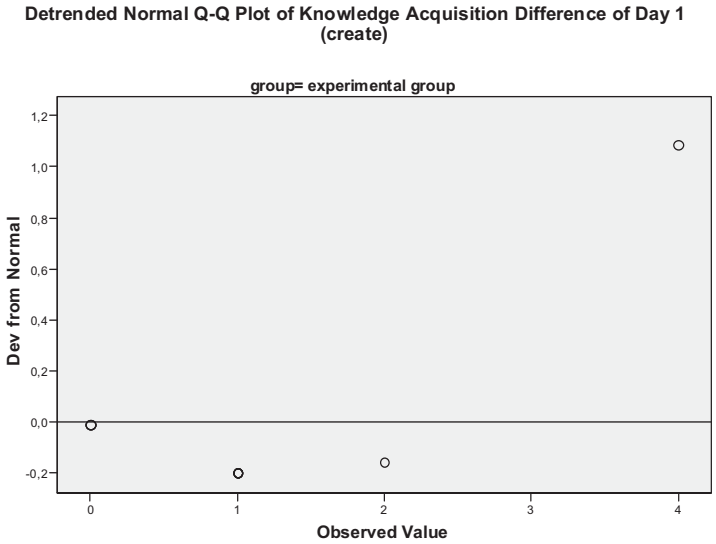


Figure 84 Detrended Q-Q plot for knowledge acquisition difference create day 1 experimental (know_diff_create)

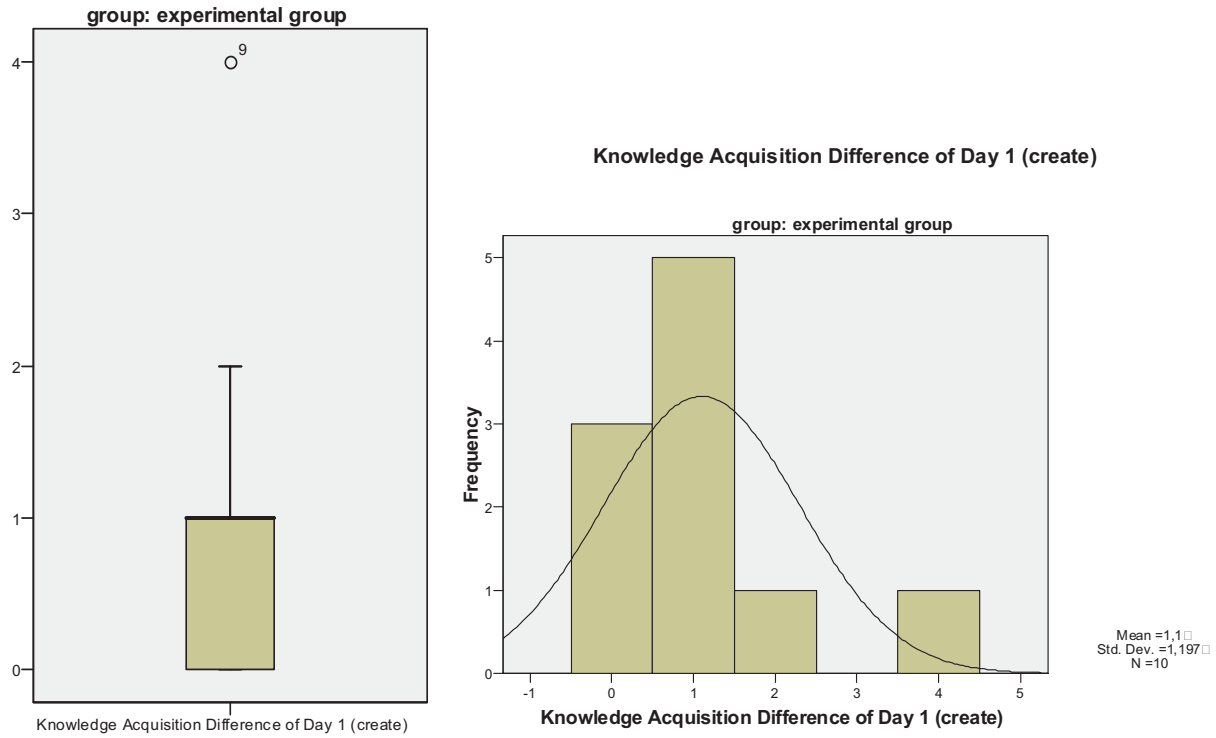


Figure 85 Histogram and boxplot for knowledge acquisition difference create day 1 experimental group (know_diff_create)

C.4.2.3 Knowledge Acquisition Difference Create Experimental Day 2

Only three different values were available for the experiment group during the second day, which makes it impossible to detect a pattern/data cluster in the detrended Q-Q-Plot. The Q-Q plot shows a strong deviation from normality for two values. In fact, know_diff_create cannot be considered as a normally distributed variable for the data set experimental group/Day 2. The reason for this is depicted in the histogram and boxplot, which show on the one side a skewness of 0.717 and a large deviation from the calculated normal curve.

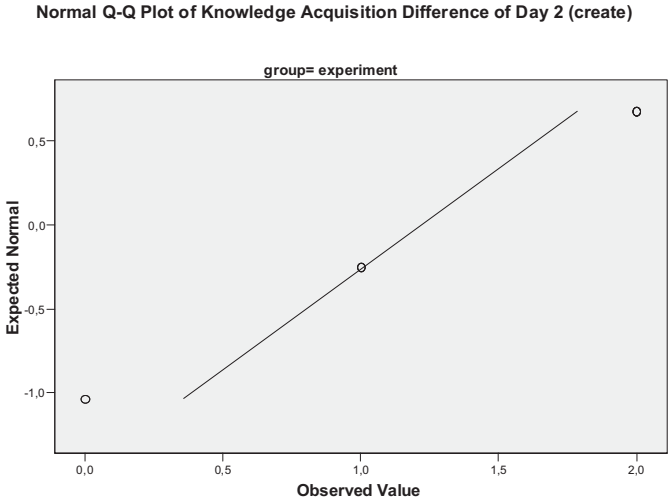


Figure 86 Q-Q plot for knowledge acquisition difference create day 2 experimental group (know_diff_create)

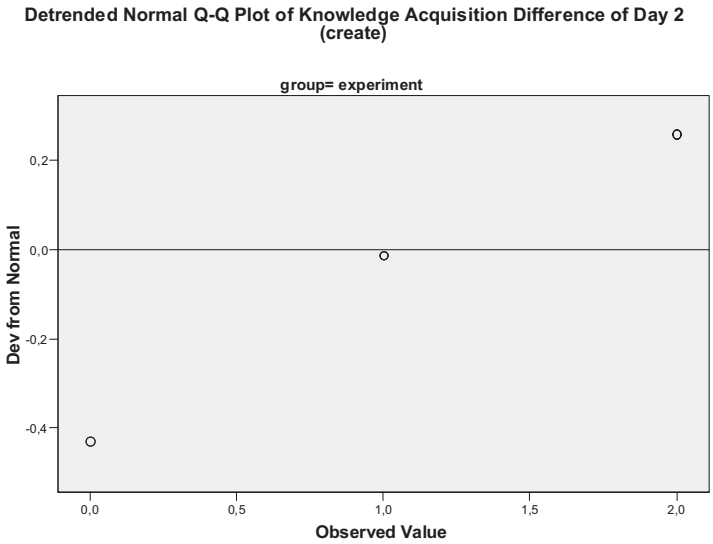


Figure 87 Detrended Q-Q plot for knowledge acquisition difference create day 2 experimental (know_diff_create)

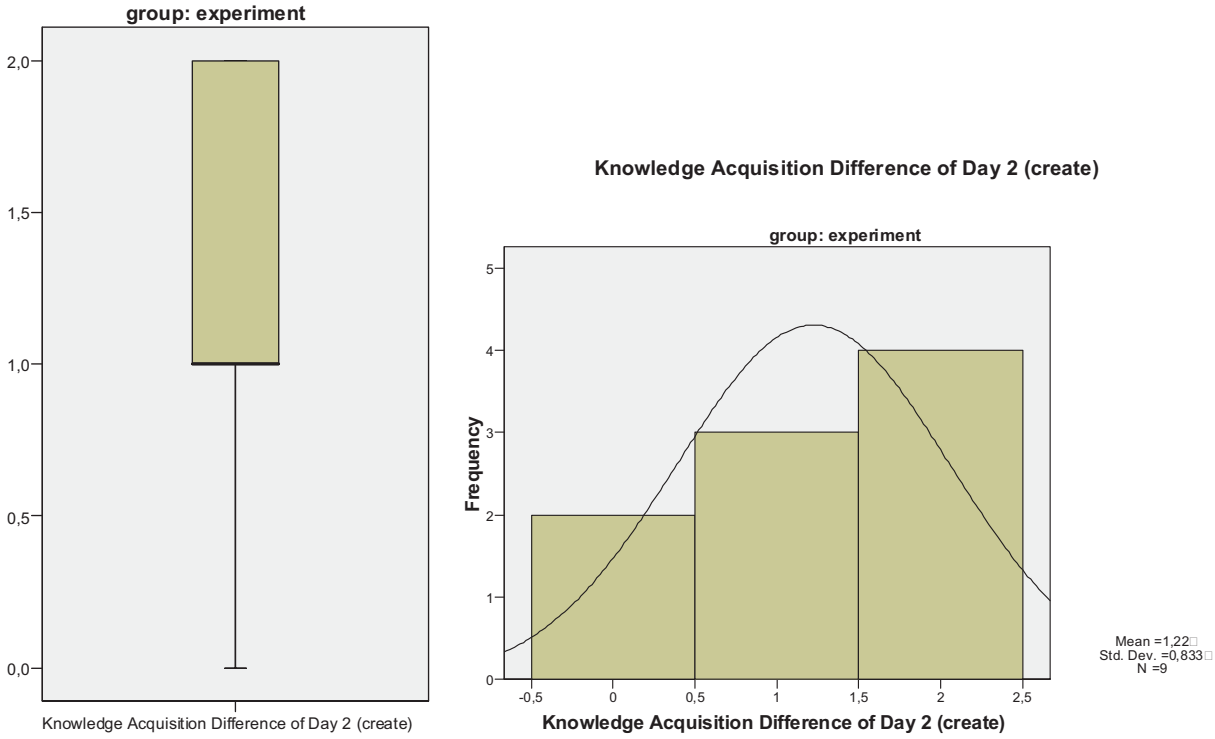


Figure 88 Histogram and boxplot for knowledge acquisition difference create day 2 experimental group (know_diff_create)

C.4.2.4 Knowledge Acquisition Difference Analyze (know_diff_analyze)

The Q-Q plot shows for the data set a high deviation from normality. In addition, the detrended normal Q-Q plot also shows higher deviations from normal and a linear pattern could be detected. Therefore, know_diff_analyze cannot be considered as a normally distributed variable. This is confirmed by the histogram and boxplot, which show on the one side a high skewness of -1.044 (i.e., a situation's asymmetry in rela-

tion to a normal distribution) and a large deviation from the calculated normal curve.

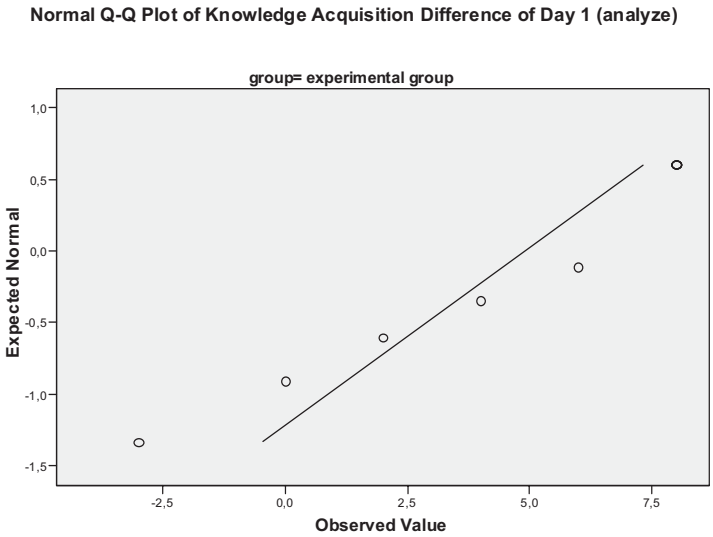


Figure 89 Q-Q plot for knowledge acquisition difference analyze day 1 experimental group (know_diff_create)

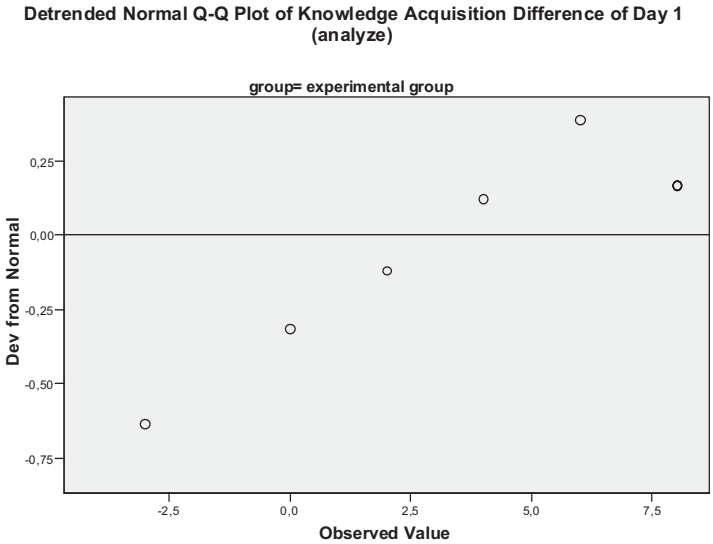


Figure 90 Detrended Q-Q plot for knowledge acquisition difference analyze day 1 experimental (know_diff_create)

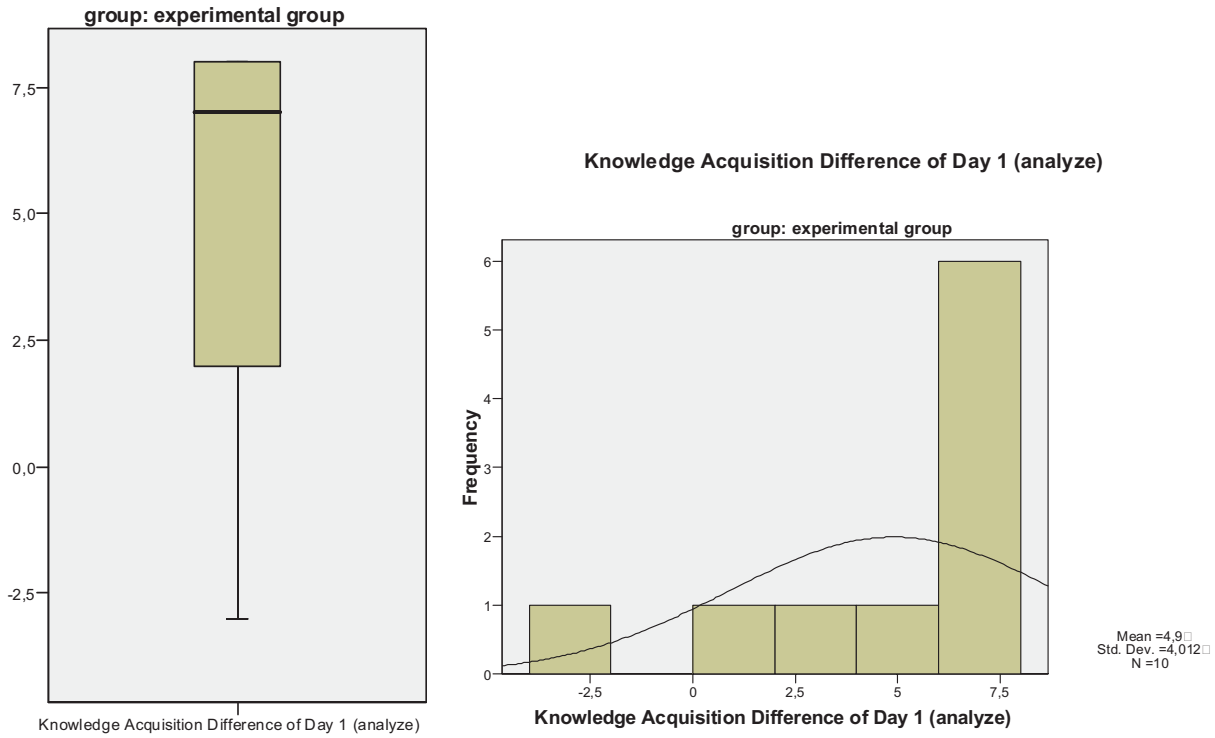


Figure 91 Histogram and boxplot for knowledge acquisition difference analyze day 1 experimental group (know_diff_create)

Hence, for the dependent variables analyzed in Section C.4.1.3, no parametric methods can be used or should be used only carefully and only when a non-parametric method is used in addition for checking the outcome of the parametric test.

C.4.3 Dependent Variables (based on period differences)

The following table shows the outcome of the Shapiro-Wilk test and the Kolmogorov-Smirnov test for period differences. A normal distribution is a prerequisite for applying dependent sample hypothesis tests. The outliers were removed from the data sets before the normality test was performed.

Table 99 Test for normality for dependent variables (based on period differences)

	Dependent Variables	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Understanding correctness	ucorr	.188	18	.093	.879	18	***.025***
Knowledge Acquisition Difference	know_diff	.136	19	.200(*)	.953	19	.446

	Dependent Variables	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Knowledge Acquisition Difference (remember)	know_diff_remember	.125	16	.200(*)	.969	16	.823
Knowledge Acquisition Difference (understand)	know_diff_understand	.196	19	.053	.897	19	***.043***
Knowledge Acquisition (apply)	know_diff_apply	.109	18	.200(*)	.966	18	.718
Knowledge Acquisition Difference (analyze)	know_diff_analyze	.099	19	.200(*)	.950	19	.396
Knowledge Acquisition Difference (create)	know_diff_create	.223	18	.018	.867	18	***.016***
Application Efficiency (based on score of marks and refactoring)	aeff	.137	18	.200(*)	.986	18	.991
Application Completeness based on score of marks and refactoring)	acomp	.109	18	.200(*)	.966	18	.718
Application Accuracy	aaccu	.126	19	.200(*)	.977	19	.909

* This is a lower bound of the true significance. a Lilliefors Significance Correction

It can be seen that for the dependent variables ucorr, know_diff_understand, and know_diff_create, the p-value of the Shapiro-Wilk test is not higher than 0.05, which means that it can be assumed that these factors are not normally distributed. The following figures show the histograms, boxplots, and (detrended) Q-Q plots for these dependent variables to further analysis of further the deviations from a normal distribution.

The chart will show some outliers based on their values for the period differences. Outliers were already considered based on the data of the two groups and the two periods. Therefore, no additional outlier analysis was performed for the period differences.

C.4.3.1 Understanding Correctness (ucorr)

The Q-Q plot shows a deviation from normality for several values and the outlier value in particular. A pattern/data cluster in the detrended Q-Q plot could be detected as well, which confirms the deviation from a

normally distribution. In fact, ucorr cannot be considered as a normal distributed variable. The reason for this is depicted in the histogram and boxplot, which show on the one side a skewness of 1.508 and the outlier value. Hence, a parametric dependent sample test should be performed with care and only in combination with a non-parametric test.

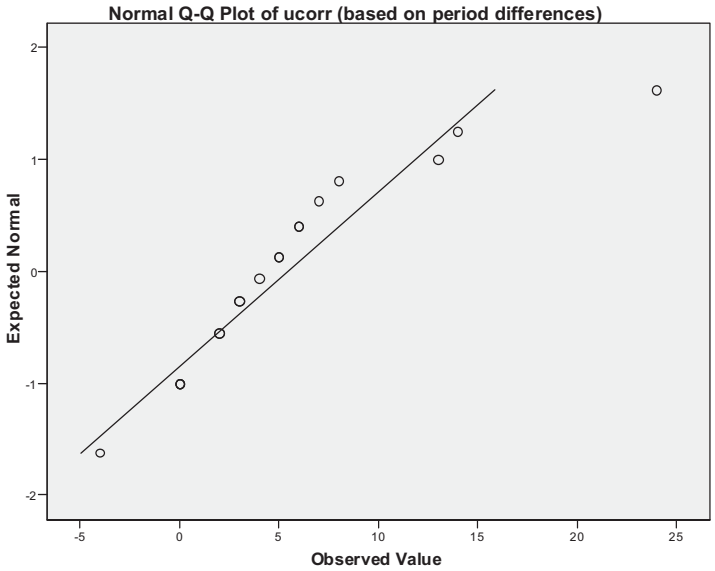


Figure 92 Q-Q plot for ucorr (based on period differences)

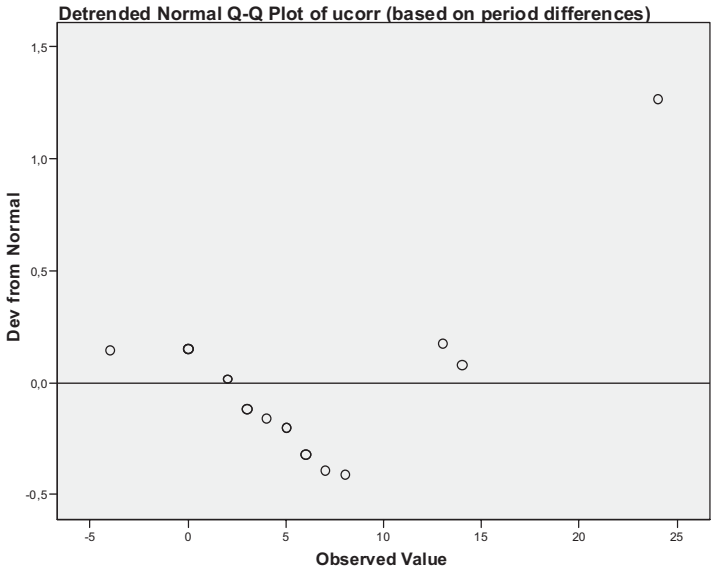


Figure 93 Detrended Q-Q lot for ucorr (based on period differences)

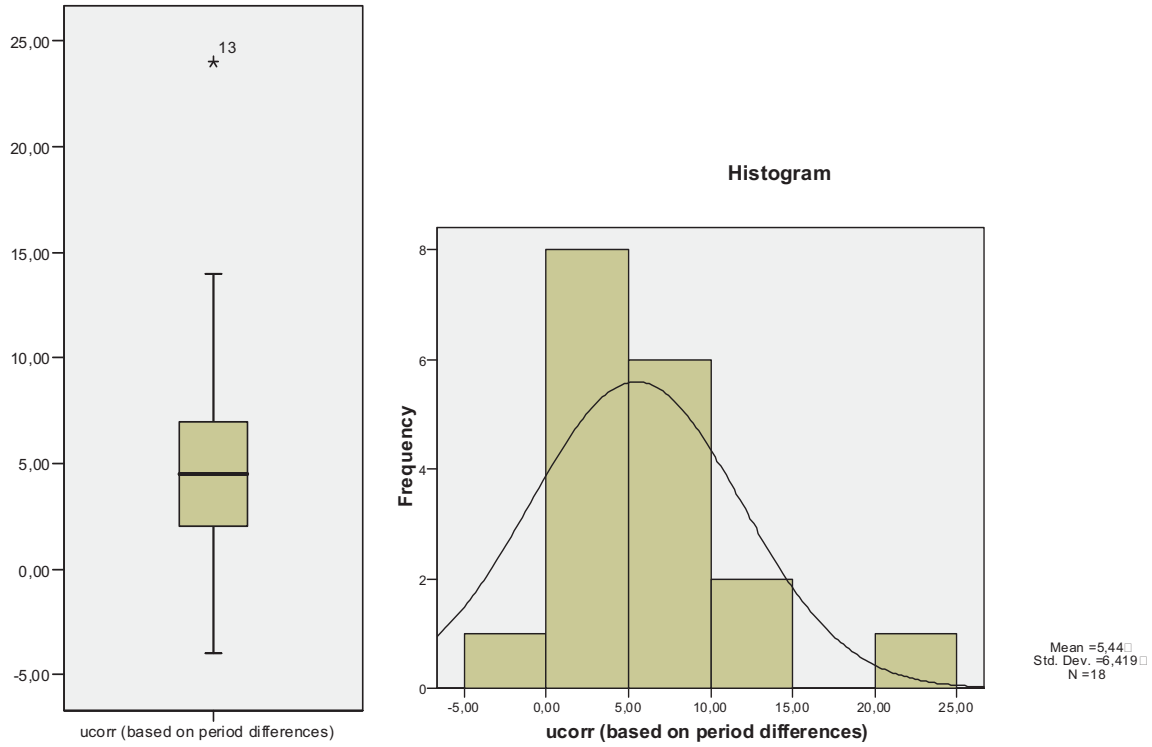


Figure 94 Histogram and boxplot for ucorr (based on period differences)

C.4.3.2 Knowledge Acquisition Difference Understand (know_diff_understand)

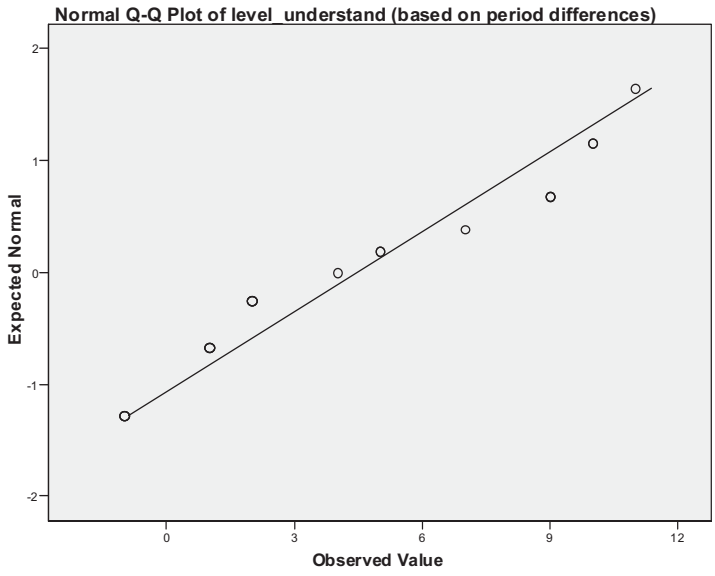


Figure 95 Q-Q plot for know_diff_understand (based on period differences)

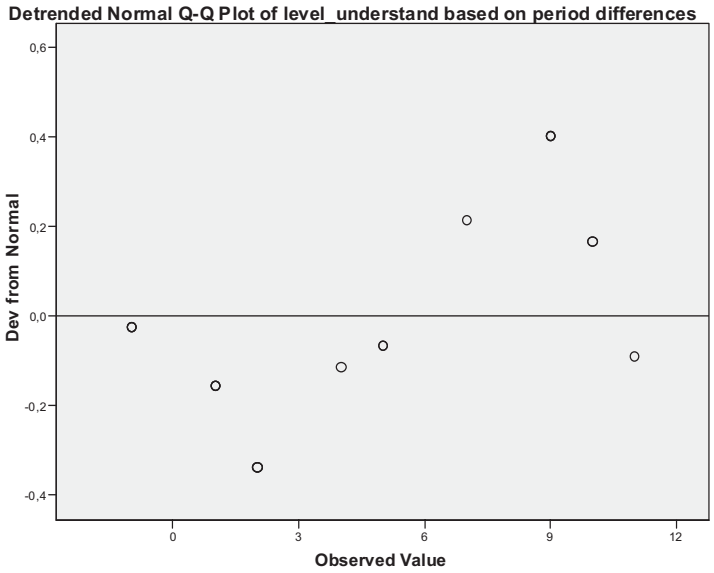


Figure 96 Detrended Q-Q plot for know_diff_understand (based on period differences)

The Q-Q plot shows a deviation from normality for several values. A pattern/data cluster in the detrended Q-Q plot could be detected as well, even if it is not so strong as for the variable ucorr (i.e., the p-value for this values was also higher). In fact, know_diff_understand cannot be considered as a normally distributed variable. The reason for this is depicted in the histogram and boxplot, which show on the one side a medium skewness of 0.524 and a high kurtosis of -1.513. Hence, a parametric dependent sample test should be performed with care and only in combination with a non-parametric test.

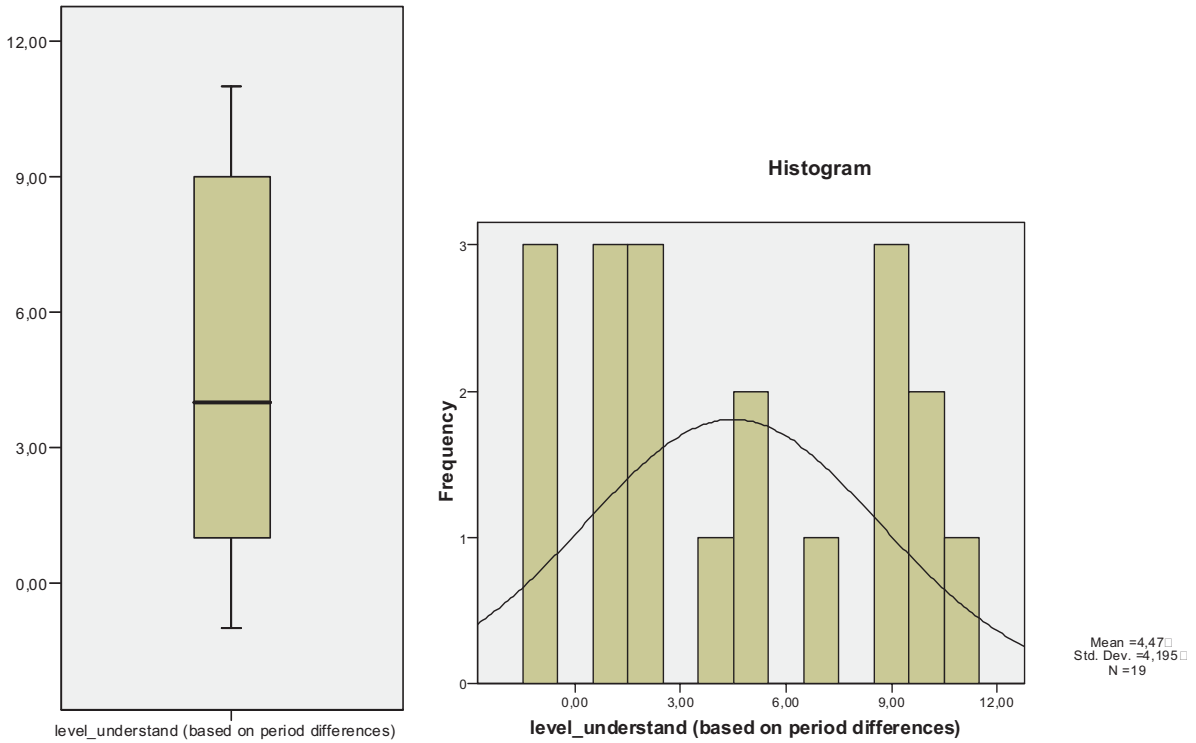


Figure 97 Histogram and boxplot for know_diff_understand (based on period differences)

C.4.3.3 Knowledge Acquisition Difference Create (know_diff_create)

Only a few different values were available for know_diff_create, which makes it difficult to interpret the plots. The Q-Q plot show a strong deviation from normality for one value. A pattern/data cluster in the detrended Q-Q plot could not really be detected. In fact, know_diff_create should not be considered as a normally distributed variable. The reason for this is depicted in the histogram and boxplot, which show a high kurtosis of 2.556 and one outlier value. Hence, a parametric dependent sample test should be performed with care and only in combination with a non-parametric test.

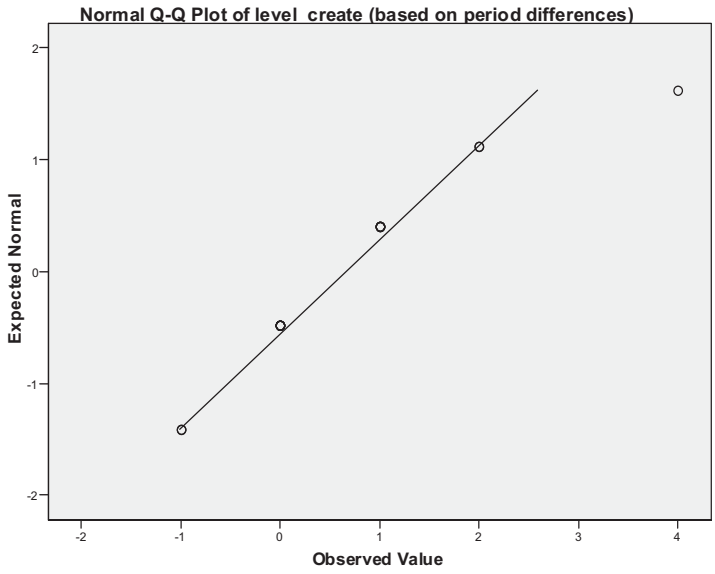


Figure 98 Q-Q plot for know_diff_create (based on period differences)

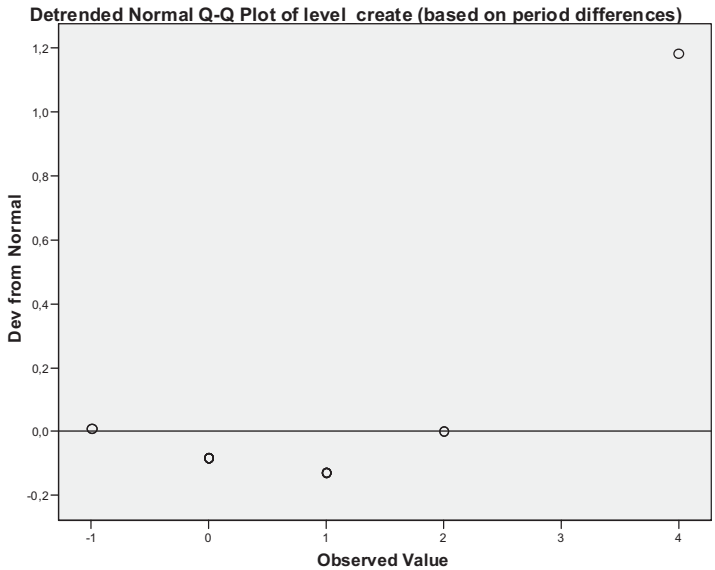


Figure 99 Detrended Q-Q plot for know_diff_create (based on period differences)

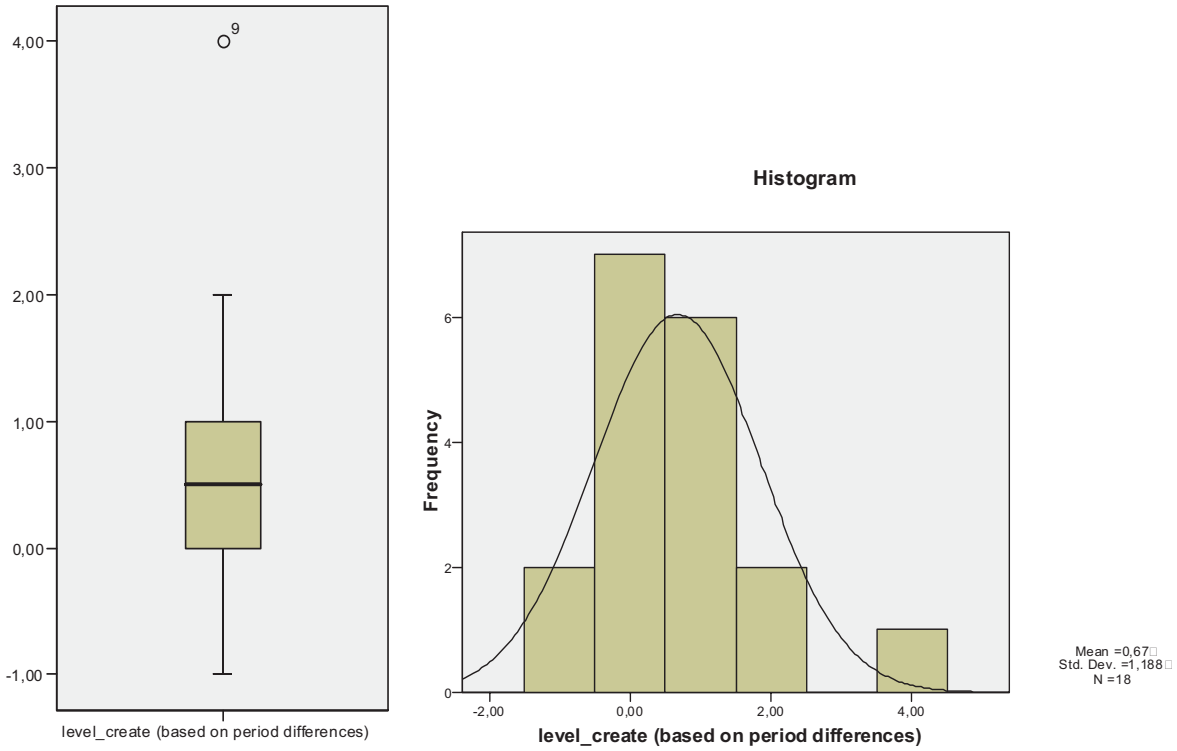


Figure 100 Histogram and boxplot for know_diff_create (based on period differences)

C.4.4 Dependent Variables (based on sequence totals)

The following table shows the outcome of the Shapiro-Wilk test and the Kolmogorov-Smirnov test for sequence totals, which is used to test for carry-over effects. A normal distribution is a prerequisite for applying independent sample hypothesis tests. The outliers were removed from the data sets before the normality test was performed.

Table 100 Test for normality for dependent variables (based sequence totals)

	Dependent Variables	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Understanding correctness	ucorr	.134	18	.200(*)	.942	18	.319
Knowledge Acquisition Difference	know_diff	.087	19	.200(*)	.973	19	.827
Knowledge Acquisition Difference (remember)	know_diff_remember	.135	16	.200(*)	.976	16	.920
Knowledge Acquisition Difference (understand)	know_diff_understand	.155	19	.200(*)	.940	19	.263

	Dependent Variables	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
Knowledge Acquisition (apply)	know_diff_apply	.113	18	.200(*)	.964	18	.679
Knowledge Acquisition Difference (analyze)	know_diff_analyze	.139	19	.200(*)	.958	19	.532
Knowledge Acquisition Difference (create)	know_diff_create	.147	18	.200(*)	.942	18	.319
Application Efficiency (based on score of marks and refactoring)	aeff	.119	18	.200(*)	.960	18	.604
Application Completeness based on score of marks and refactoring)	acomp	.113	18	.200(*)	.964	18	.679
Application Accuracy	aaccu	.113	19	.200(*)	.950	19	.401

* This is a lower bound of the true significance. a Lilliefors Significance Correction

It can be seen that for all dependent variables, the totals for both sequences do not significantly deviate from a normal distribution. Hence, a parametric test for carry-over testing can be applied.

C.5. Analyzing Confounding Effects

C.5.1 Terminology

Frequently, different terms for the same effect are used – this makes a comparison of statistical approaches extremely difficult. Therefore, this section will first clarify the chaos in terminology starting with a first classification taken from clinical research. The subsequent sections will provide approaches to the investigation and/or correction of the effects confounding the treatment effect, if such effects exist.

Senn's classification of confounding effects

Senn states that when analyzing the data from crossover trials, we expect that the crossover differences are distributed at random around the true treatment effect. However, Senn names several factors that might cause the crossover differences not to be distributed at random across the true treatment effect in crossover studies in clinical research (Senn, 1993). The following list describes the factors according to Senn and will provide a basis for classifying effects confounding the treatment effect:

Period effect: A trend that might affect the experiment as a whole. A period effect is also called a trend effect, since it is an effect related to the mean cross-offer difference between the two sequences (e.g., noisy environment during the whole experiment, which impacts both sequences; different experience levels and background between groups/sequences; drug tolerance or resistance). A period effect is a change that would have occurred even in the absence of treatment.

Period by treatment interaction: This effect is related to the fact that the treatment effect varies according to the period in which is was given. (e.g., the patients might be affected by hay fever on the first visit but not on the second one. This might lead to a period effect, but if one of the treatments were effective for asthma in general, except when complicated by or provoked by hay fever, this would also lead to a treatment by period interaction).

Carry-over effect: Carry-over effects, or *residual effects*, are effects of a treatment that persist after the end of the treatment period; i.e., the response to a current treatment is affected by what treatment was applied in a previous period. A carry-over effect will bias the estimation of the treatment effect (e.g., learning effect after the first treatment).

Patient by treatment interaction: This effect occurs when there is no general treatment effect, but the effect varies from subject to subject. This effect cannot be investigated in a two-period crossover experiment; treatments need to be applied to the subjects a number of times.

Patient by period interaction: This effect would arise if subjects were confronted with period effects that were not the same for all subjects (e.g., some subjects have been in a noisy environment or some subjects are suffering from hay fever and the others are not).

The following table shows Senn's classification in the first row. The other rows relate the terminology of other authors to the classification of Senn. In addition, the term *sequence effect* has been added, since it is found very often in the literature (mostly in books on basic statistics and other domains such as human sciences). Reed, for example, defines a sequence effect as an effect due to the different orders in which treatments are given (Reed, 2004). The same is also stated by Bortz and Döring, who add the term "Positionseffekt", respectively "Kontexteffekt", which is an interaction effect between treatment and sequence (i.e., the treatment effect depends on the position in a sequence) (Bortz & Döring, 2001). These different orders of treatments are not explicitly mentioned by Senn, but need to be included.

Table 101 Overview of confounding effects

	Period Effect	Period by Treatment Interaction	Carry-Over Effect (residual effect)	Sequence Effect	Patient by Treatment Interaction	Patient by Period Interaction
(Senn, 1993)	**x**	**x**	**x**		**x**	**x**
(Reed, 2004)	x	period effect	x	x		
(Jones & Kenward, 2003)	x	direct effect by period interaction	x			
(Armitage & Berry, 1994)		x	x			
(Grieve, 1985)	x	carry-over effect used as synonym	x			
(Kirk, 1995)		x	x	carry-over effect due to different treatment orders		
(Winer et al., 1991)			x	or order effects		
(Bortz & Döring, 2001)				*Kontext-* or. *Positionseffekt*		
(Bortz, 2005)			*Sequentieller Übertragungseffekt*			
(Díaz-Uriarte, 2002)	x		x	Sequence effect or group main effect		
(Kitchenham, Fry, & Linkman, 2003)	x	"mostly" carry- over effect	Same as period by treatment interaction			
(Juristo & Moreno, 2001)				related to learning effect		

Reasons why confounding effects are not easily separable

The inconsistent usage of the terms is due to the fact that the effects are not separable from each other, either because they are a subtype of another effect type, because they can only occur when another effect exists, because the experiment design does not allow a distinction of the effects, or because they are just understood wrongly. Reed, for example, does not distinguish between period effects and period by treatment interaction – according to the classification by Senn, Reed is talking about period by treatment interaction (Reed, 2004). Kitchenham et al. state that period by treatment interaction happens when the effect of the treatment differs according to the order in which the treatment occurs and is usually restricted to carry-over effects from the preceding drug (Kitchenham et al., 2003). Furthermore, Kitchenham et al. state that

when subjects improve their performance during the course of a sequence of tasks, this results in a period effect – this is simply an example of a wrong statement. The description by Kitchenham et al. corresponds to the description of "Positionseffekt" by Bortz and Döring, who state that this effect is an interaction effect between treatment and sequence. Jones and Kenward report that different amounts of carry-over effects from the treatments can be the cause of period by treatment interaction effects. Kirk state that the portion of carry-over effects that is attributable to the order of treatments is referred to as sequence effects (Kirk, 1995). Reed says that unlike sequence effects, carry-over effects affect the treatment response only in the second time period (Reed, 2004). Juristo and Moreno state that learning effects are often detected by considering the order in which the experiments have been performed as a factor and designing a factorial together with the principal factor under examination (Juristo & Moreno, 2001).

Other authors use different terms than the ones proposed by Senn: e.g., Reed captures the period by treatment interaction effect under the term *period effect* (Reed, 2004). Díaz-Uriarte calls the sequence effect *group main effect* (Díaz-Uriarte, 2002).

The question is which kind of effects may influence the estimation of the treatment effects in this experiment? Seen states that the first effect (i.e., period effect) can be easily dealt with by adjusting the treatment effect by applying specific methods (see below). Jones and Kenward state that the chance of treatment interacting with period (i.e., period by treatment interaction) will be small in well-planned experiments. Since we kept the working environment of the experiment constant over the two subsequent days, there is no need to expect such an effect in this experiment. The patient by treatment interaction and patient by period interaction do not cause much of a problem because "they only impact the general variability of the results and only may cause difficulties with interpretation" (Senn, 1993). Furthermore, the simple within-subject design does not allow the investigation of the patient by treatment interaction effect. More complex designs would be required.

Approaches for detecting and/or correctng confounding effects

The following table shows the most relevant work by researchers who proposed approaches to detecting and/or correcting the confounding effects. The first row lists the effects that should be considered for discussion or statistical investigation from the perspective of this experiment because they might exist due to the experiment's design.

Table 102 Confounding effects in a counterbalanced, within-subject design

	Period Effect (π)	**Sequence Effect**	**Period by Treatment Interaction**	**Carry-over Effect (λ)**
Relevancy for this experiment	**yes**	**Yes, but expected to be small due to randomization**	**Yes, but expected to be small**	**yes**
(Senn, 1993)	- two-sample t-test: Testing for the inequality of period effects by a two-sample t-test with period differences between the two sequences Adjusting for a period effect: Apply the Hill Armitage approach (comparing period differences between the two sequence groups)	-	- test of equality of carry-over by using a two sample t-test based on differences between subject totals ; however, the test has low power	
(Reed, 2004)	-	-	2-stage procedure with testing three hypotheses: CROSS (equality of treatment and carry-over effect), SEQ (was the rejection of CROSS due to treatment difference or to carry-over effects?), PAR (t-test applied to first period only when significant carry-over effects have been found)	
(Jones & Kenward, 2003)	- test of inequality of period effects, assuming that there are no carry-over effects → two sample t-test based on crossover differences - Analysis of variance (split-plot ANOVA)	-	-	- test of equality of carry-over by using a two-sample t-test based on differences between subject totals; however, the test has low power - Analysis of variance (split-plot ANOVA)
(Grieve, 1985)	-	-	Baysesian analysis based on the Bayes factor against unequal carry-over effects	

	Period Effect (π)	Sequence Effect	Period by Treatment Interaction	Carry-over Effect (λ)
(Kirk, 1995)	-	Apply randomzation to pervent the sequence effect → no statistical procedure provided	-	-
(Winer et al., 1991)	-	- Apply randomization to prevent sequence effects from being completely confounded with one or just a selected few of the treatments → no statistical procedure provided - In order to control sequence effects: use the latin-square design principle by introducing a factor with alternatives that correspond to the sequences	-	-
(Bortz & Döring, 2001) & (Bortz, 2005)	--	In order to control sequence effects: use the latin-square design principle by introducing a factor with alternatives for each sequence	-	-
(Díaz-Uriarte, 2002)	- Test for inequality of period effect: two-sample t-test by using the crossover differences (differences between period 1 and period 2 for subjects in AB, and differences between period 2 and period 1 for subjects in BA) - Adjusting for the period effect: Hill-Armitage approach			

A 2x2 experiment does not allow the separation of all confounding effects

The next sections will investigate whether carry-over, sequence (and position), or period effects exist, and how this will impact the further analysis of this experiment. However, it must be mentioned that the 2x2 design does not allow to completely distinguish between the effects

when they have been detected. For example, the test for carry-over effects requires assuming that there are no sequence effects because in the design matrix, the columns for sequence effects and differential carry-over effects are identical (Díaz-Uriarte, 2002) (a more detailed explanation of this fact is given in Section C.5.3).

First, sequence effects will be investigated in Section C.5.2. If no sequence effects are detected, carry-over effects can be investigated in Section C.5.3. The last Section C.5.4 investigates whether there are any period effects.

C.5.2 Investigation of Sequence Effects and Treatment*Sequence Interaction Effects

Sequence effects were investigated in this experiment by using the sequence factor

A sequence effect is an effect due to the different orders in which treatments are given. Randomizing the order of treatments independently for each subject is an effective way of reducing sequence effects. Another alternative way of controlling sequence effects is to include the effects as one of the treatments in the design (Kirk, 1995). Therefore, an additional factor called *sequence* was added to the design of the experiment. Designs of this kind in software engineering have been applied, for example, in (Daly, Brooks, Miller, Roper, & Wood, 1995) and (Macdonald & Miller, 1998).

An analysis of variance (ANOVA) for repeated measures was performed separately for each of the dependent variables with $\alpha = 0.05$ (i.e., a general linear model for repeated measures Type III was used because the cells contain a different number of cases – it was unbalanced regarding cell frequencies) without the outliers. The ANOVA was done with one within-subjects factor *treatment* (LSEP | EP) and one between-subjects factor *sequence* (LSEP→EP | EP→LSEP). *Sequence*treatment* consists of an interaction effect between the factors' treatment and the sequence, i.e., a position effect, which means that the treatment effect depends on the position in a particular sequence. No covariates, i.e., disturbing factors, were considered in this analysis step in order to keep the interpretation of the results simple.

Using ANOVA for testing of sequence effects and treatment*sequence interaction effects

The following tables show the F-Value, p-value, partial eta squared, and power of the repeated measures ANOVA. The partial eta squared statistic (i.e., the effect size of the related effect) reports the "practical" significance of each term, based upon the ratio of the variation (sum of squares) accounted for by the factor to the sum of the variation accounted for by the factor and the variation left to error, i.e., the eta-squared statistic describes the proportion of total variability attributable to a factor. If a partial eta squared term is near 0, this shows that it accounts for a negligible amount of variation compared to the error term. Levene's test of equality of error variances was not significant for all vari-

ables, which means that the variances are not significantly different and hence, that the ANOVA can be applied.

In the following, only the tables for those variables are shown where a position effect or a sequence effect was detected. Profile plots illustrate the interaction between sequence and treatment (i.e., position effect). Mauchly's test (i.e., testing the null hypothesis of sphericity) was not relevant because the degree of freedom was zero. Hence, the assumption of sphericity was not relevant and ANOVA could be performed without performing this test. *Intercept* is not of interest for us, but it has been kept in the between-subjects analysis table for reasons of completeness. The intercept row checks the hypothesis that the grand mean (i.e., the mean of all cell values) is zero.

Table 103 Test of within-subjects effects for understanding completeness (ucorr)

Source	Type III Sum of Squares	df	Mean Square	F	p-value	Partial Eta Squared	Power
Treatment	301.606	1	301.606	19.621	.000	.551	.986
Sequence * Treatment	104.272	1	104.272	6.783	***.019***	.298	***.687***
Error	245.950	16	15.372				

Table 104 Test of between-subjects effects for understanding completeness (ucorr)

Source	Type III Sum of Squares	df	Mean Square	F	p-value	Partial Eta Squared	Power
Intercept	44023.472	1	44023.472	755.160	.000	.979	1.000
Sequence	1.250	1	1.250	.021	.885	.001	.052
Error	932.750	16	58.297				

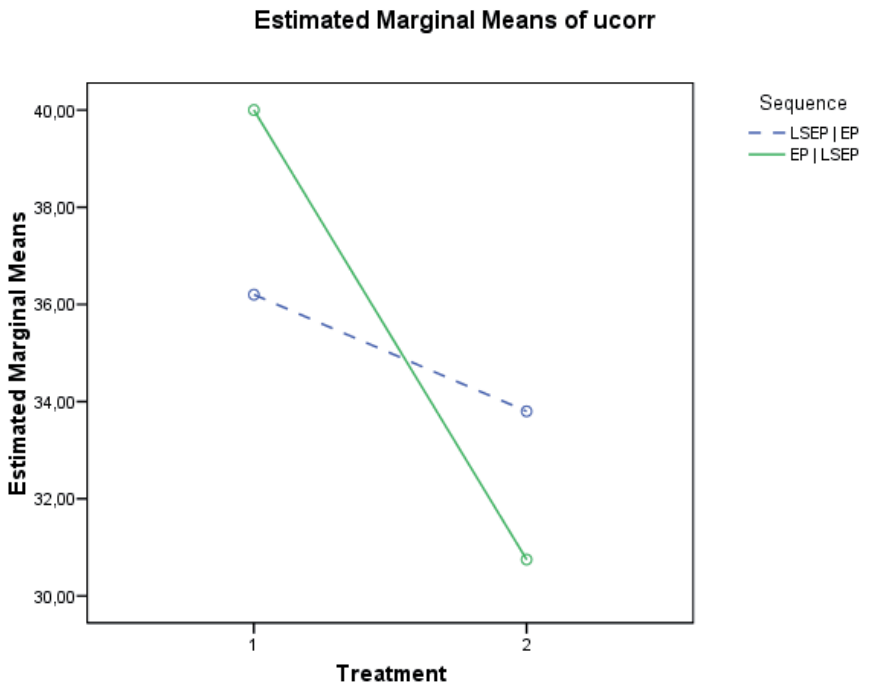


Figure 101 Profile plot for understanding correctness (ucorr)

Position effect = treatment * sequence interaction effect

For understanding correctness, a position effect (i.e., treatment*sequence interaction) effect was detected (p-value=.019). The profile shows a crossing of both lines, which is an indication of an interaction effect. However, the power is only .687, which means that we cannot confirm the existence of any significant interaction between treatment and sequence. A sequence effect could not be detected at all.

Table 105 Test of within-subjects effects for knowledge acquisition difference apply (know_diff_apply)

Source	Type III Sum of Squares	df	Mean Square	F	p-value	Partial Eta Squared	Power
Treatment	141.016	1	141.016	8.914	.009	.358	.800
Sequence * Treatment	155.210	1	155.210	9.811	***.006***	.380	***.836***
Error	253.118	16	15.820				

Table 106 Test of between-subjects effects for knowledge acquisition difference apply (know_diff_apply)

Source	Type III Sum of Squares	df	Mean Square	F	p-value	Partial Eta Squared	Power
Intercept	3911.460	1	3911.460	182.013	.000	.919	1.000
Sequence	6.043	1	6.043	.281	.603	.017	.079
Error	343.840	16	21.490				

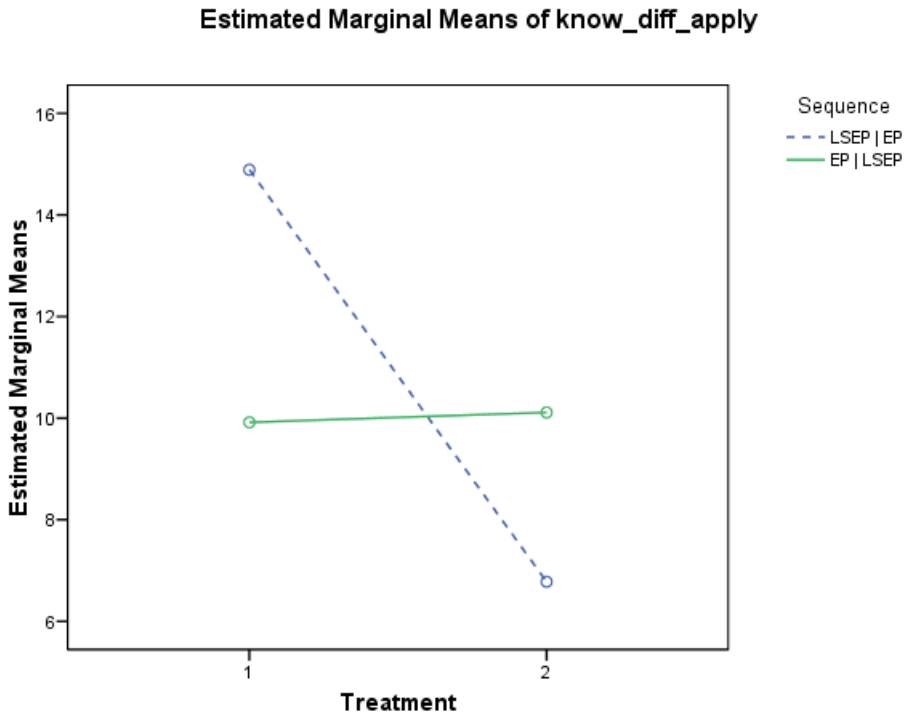


Figure 102 Profile plot for knowledge acquisition difference apply (know_diff_apply)

For know_diff_apply, a significant interaction effect (p-value=.006) was detected with a power of .836. In this case, we have to confirm the existence of a position effect between treatment and sequence.

Table 107 Repeated measures ANOVA for sequence effect test for application completeness (acomp)

Source	Type III Sum of Squares	df	Mean Square	F	p-value	Partial Eta Squared	Power
Treatment	.245	1	.245	8.914	.009	.358	.800
Sequence * Treatment	.269	1	.269	9.811	***.006***	.380	***.836***
Error	.439	16	.027				

Table 108 Test of between-subjects effects for test for application completeness (acomp)

Source	Type III Sum of Squares	df	Mean Square	F	p-value	Partial Eta Squared	Power
Intercept	6.791	1	6.791	182.013	.000	.919	1.000
Sequence	.010	1	.010	.281	.603	.017	.079
Error	.597	16	.037				

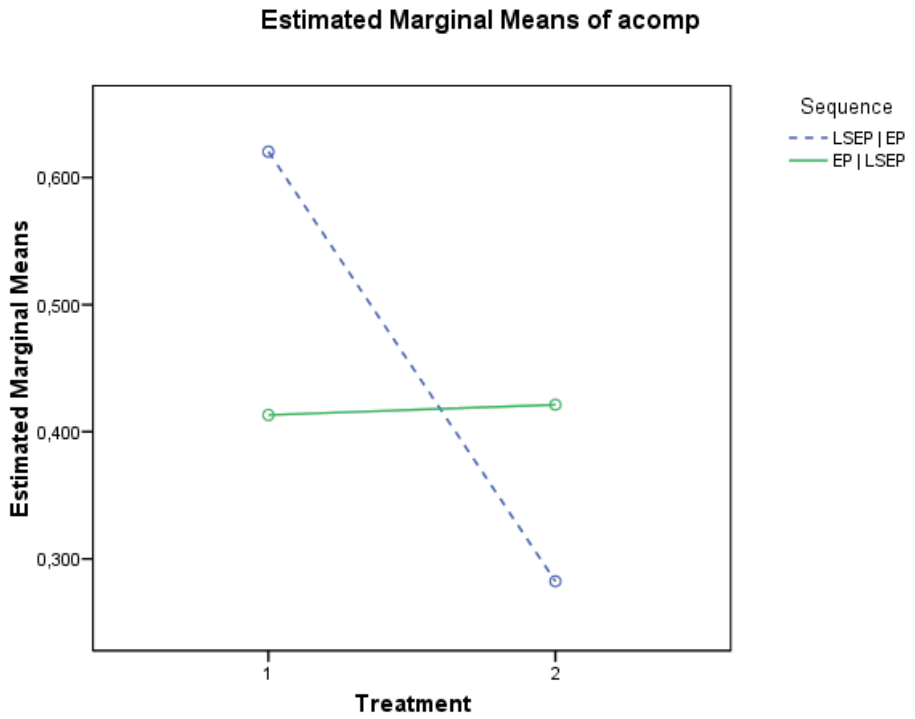


Figure 103 Profile plot for test for application completeness (acomp)

One significant position effect was detected

A significant position effect was only detected for the variables knowledge difference acquisition apply (know_diff_apply) and application completeness (acomp). The reason why the p-values, power, and effect size are the same is that both variables are based on the same data (i.e., know_diff_apply is calculated based on the refactoring scores; acomp also uses this score). They are not due to any difference in the experience level of the two groups (this would have resulted in a period effect), but could be related to the different experience packages provided during the two periods or to a fatigue effect during the second day. Section C.5.2 shows how position effects can be corrected.

No sequence effects were detected

It is important that no sequence effects exist, because carry-over effects cannot be distinguished from sequence effects in these kinds of experiments.

C.5.3 Investigating Carry-Over Effects and Period by Treatment Effects

Carry-over effect

Greenwald states that a common risk of applying a within-subjects design is the existence of carry-over effects between the periods (Greenwald, 1976). Carry-over effects can be effects due to a treatment that persist after the end of the treatment period and influence the subsequent treatment. In this experiment, the subjects could earn practical experiences in the domain of refactoring, which they did not possess before the experiment. This could lead to higher performance during the second period than during the first period. Many researchers from the domain of human or animal research, such as Díaz-Uriarte (Díaz-Uriarte,

2002) or (Abeyasekera & Curnow, 1984), have shown that counterbalancing and randomization cannot take care of carry-over effects.

Approaches for carry-over effect testing are highly criticized

Hence, carry-over effects may still exist and may bias the conclusions made based on the statistical analysis. Several approaches exist to test for carry-over effects. However, the applied carry-over tests have often been criticized. Several experiments in software engineering have investigated carry-over effects by using the approach by (Grizzle, 1965) and (Mills & Armitage, 1979), which compares the variability between sequences with the variability of subjects within sequences. (e.g., applied by (Laitenberger, 2000)). This two-stage analysis procedure is now known to be extremely biased and is not recommended because the test has low power. Freeman showed that Grizzle's two-stage procedure of testing for carry-over difference in the first stage and then for a direct treatment difference in the second stage not only inflates the probability of making a type I error, but also produces a biased estimate of the direct treatment difference (Freeman, 1989). Therefore, the conclusions are questionable: "The lack of effects reported in some studies could be the consequence of inflated variances, and the significant effects reported in others could be the result of either period or carry-over effects (Díaz-Uriarte, 2002)". In medicine, one way of diminishing the impact of carry-over effects is the incorporation of lengthy washout periods in the experimental design. A washout period is defined as the time between the treatments. Instead of immediately stopping and then starting the new treatment, there will be a period of time where the treatment from the first period, i.e., the drug, is washed out of the patient's system. However, these washout periods are not applicable to experiments in software engineering, since carry-over effects are often related to practicing and even long "washout periods" cannot assure that the practicing effects disappear. In addition, the risk of obtaining period effects increases when the time between the treatments increases.

Period by treatment interaction effect

Another more general type of carry-over effect is the period by treatment interaction effect, which is due to *secular change*, i.e., some factor other than the treatment might slowly be affecting the condition of the subjects and the benefit of one treatment compared to another might be dependent on the current state of the subject. One problem with AB|BA crossover designs is that it is not possible to separately distinguish between carry-over effects and period by treatment interaction effects (Senn, 1993). In addition, the test for carry-over effects requires assuming that there are no sequence effects. The reason for this is that one cannot estimate both carry-over effect and sequence effect, because the 2x2 design yields only four cells means, and therefore only a maximum of four parameters can be estimated: overall grand mean, the treatment effect, the period effect, and the fourth parameter, which is either the sequence effect of the carry-over effect, but not both.

Not the existence of carry-over effects is critical per se, but the differential carry-over effects

In fact, it is not the presence of carry-over effects per se that leads to aliasing with direct treatment effects in the (LS|EP) crossover, but rather the existence of differential carry-over effects, i.e., the carry-over effect due to treatment LS differs from the carry-over effect due to treatment EP. If the carry-over effects for EP and LS are equivalent in the LS|EP within-subjects design, then this common carry-over effect is not aliased with the treatment difference. A test for carry-over effects can be done by using the difference between the subjects' totals in the two periods between the two sequences. When using the totals, the differences cannot differ by a treatment effect, because each subject had both treatments, nor can they differ by any period effect, because each subject was treated in both periods. If, however, a treatment effect persists, then in the second period, the subject in the LSEP|EP sequence will have a carry-over effect from LSEP, whereas the subject in the EP|LSEP sequence will have a carry-over effect from EP. The totals of the variables do not deviate significantly from a normal distribution (see Appendix C.4.4) – this is a prerequisite for conducting this test. The results of the independent sample t-test are provided in the following table.

Table 109 Independent sample t-test for carry-over effect and period by treatment interaction testing

	Levene's test for equality of variances		**Test for carry-over effect (λ) and period by treatment interaction effect**					
	F	**p-value**	**t**	**df**	**p-value**	**Mean Difference**	**Std. Error Difference**	**Power**
ucorr	.046	.833	-.420	16	.680	-2.60	6.19	0.068
know_diff	.837	.373	.610	17	.550	2.23	3.66	0.089
know_diff_remember	.006	.942	-2.368	14	***.033***	-3.25	1.37	0.599
know_diff_understand	.341	.567	.660	17	.518	1.19	1.80	0.096
know_diff_apply	4.076	.061	.340	16	.739	1.25	3.68	0.061
know_diff_analyze	.009	.924	1.353	17	.194	2.88	2.13	0.247
know_diff_create	.102	.754	1.180	16	.255	.89	.75	0.199
aeff	.957	.342	-.591	16	.563	-.11	.18	0.086
acomp	4.076	.061	.340	16	.739	.05	.15	0.062
aaccu	.008	.932	-.161	17	.874	-.02	.11	0.056

The significance of Levene's test is above 0.05 for all variables, which suggests that the equal variances assumption is not violated and that the t-test can be performed. It can be seen that the null hypothesis ($\lambda 1 = \lambda 2$) can be rejected for know_diff_remember, i.e., a significant carry-over effect exists. The calculation of power confirms one of Freeman's criticisms regarding this procedure, namely, that the test has low power and that this test should therefore not be used. This test delivers the same results as the procedure of Hills and Armitage (Hills & Armitage, 1979), who provided a correction to the two-stage "all-or-nothing" procedure of Grizzle (Grizzle, 1965). The procedure was called all-or-nothing be-

cause a significant test for a differential carry-over difference to the so-called PAR test (Freeman, 1989) uses only the data from the first period. Nevertheless, the Grizzle procedure is still frequently used in software engineering.

Jones and Kenward present an approach to increase the power of these tests by using baseline measurements taken during the run-in and wash-out periods (Jones & Kenward, 2003). Different types of baseline measurements exist (i.e., before first treatment, before second treatment, and after completion of last treatment). However, in this experiment, no baseline measurements were performed.

The carry-over effect should be small compared to the treatment effect

Poloniecki and Daniel, and Barker et al. state that it is not necessary that there be no carry-over effects, rather that the carry-over effect be small in comparison to the treatment effect (Barker, News, Huitson, & Poloniecki, 1995; Poloniecki & Daniel, 1981). In addition, Jones and Lewis say that a relatively small difference in the size of carry-over effects will not seriously reduce the power of the test for a treatment difference or lead to more than a small amount of bias in the estimate of the treatment difference (Jones & Lewis, 1995). The carry-over effects in this experiment were small compared to the treatment effect differences (the calculation of the effects and confidence intervals are not provided here) and therefore, the carry-over effects were ignored for further analysis.

C.5.4 Investigating Period Effects

Many researchers have stated that randomization and counterbalancing ensure that no period effects can occur (Crowder & Hand, 1990). For example, in an AB|BA crossover study, as in this experiment, Senn says that the period effect can be ignored when the subjects are allocated completely at random to the two sequences (Senn, 1993). Nevertheless, Senn motivates that when a researcher has decided to apply randomization and counterbalancing, he is aware of the existence of period effects and that hence, these effects should be dealt with in further analysis when they are known.

When period effects exist, a matched-pair t-test cannot be used

The standard statistical test in a within-subjects design is a dependent sample t-test. However, when a period effect exists, the usage of the matched-paired t-test is not adequate. There are two reasons for this (Senn, 1993): First, if there is a period effect and there are unequal numbers of subjects in each sequence, the test and the estimate of direct treatment effects will be biased. Second, even if there are equal numbers of subjects in each sequence, we will lose power: "A period effect is a systematic trend, however, by putting together subjects from both sequences, we are ascribing this systematic variation to the random component (the error term) and the standard errors of the estimates will be inflated" (Senn, 1993). Therefore, one should explicitly check for period effects, despite randomization and counterbalancing.

A two sample t-test on period difference can be used to detect period effects

There are several approaches to investigating treatment effects while adjusting them because of period effects. One simple method described in (Jones & Kenward, 2003; Senn, 1993) applies a two-sample t-test for the period differences. Another method called the Hill-Armitage approach (Hills & Armitage, 1979) delivers the same results as the simple method, but is more generizable to more complex designs (Senn & Hildebrand, 1991). Jones and Kenward also include period testing in their analysis of the variance approach (Jones & Kenward, 2003), which can also be used for carry-over testing. But before adjusting for period effects is done in a later section, an independent sample t-test according to Senn (Senn, 1993) was performed to test for period effects. This test tests the null hypothesis of period effect equality of the two sequences ($\pi_1 = \pi_2$).

First, a plot can give a first hint about a period effect. The first graph shows the average values for both treatments for the dependent variable know_diff for both periods. The second graph helps to compare the values of the two sequences.

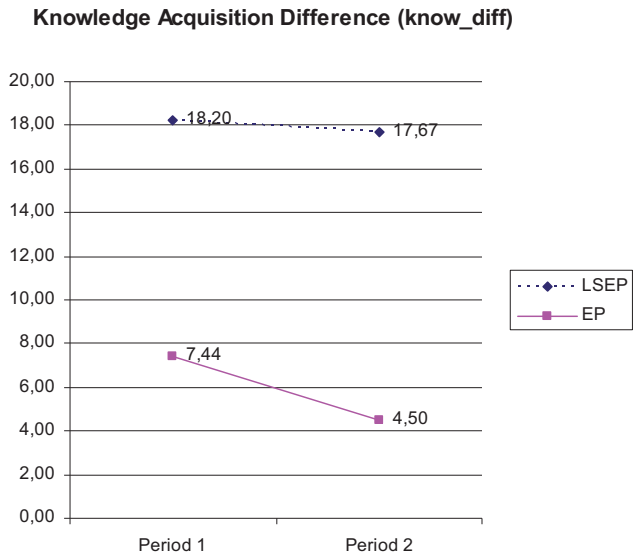


Figure 104 Plot for period effect with respect to treatment

The first graph shows that the treatment LSEP provides higher values for know_diff than the treatment EP for both periods. Furthermore, a small decrease during the second period for both treatments, which could be related to a period effect, can be observed. It is interesting to investigate whether the period effect is significantly different between the sequences or if it just affects both treatments equally. It can be seen that the average of the crossover differences differs. A two-tailed independent t-test at $\alpha = 0.05$ was applied to test whether the period effect was significant for all dependent variables. Such a test for period effects is based on crossover differences, which are subject differences between treatment LSEP and treatment EP. If a constant trend is present, then it must affect each of the period differences identically. A period

effect regarding information quality could not be investigated, since information quality was gathered only at the end of the second period.

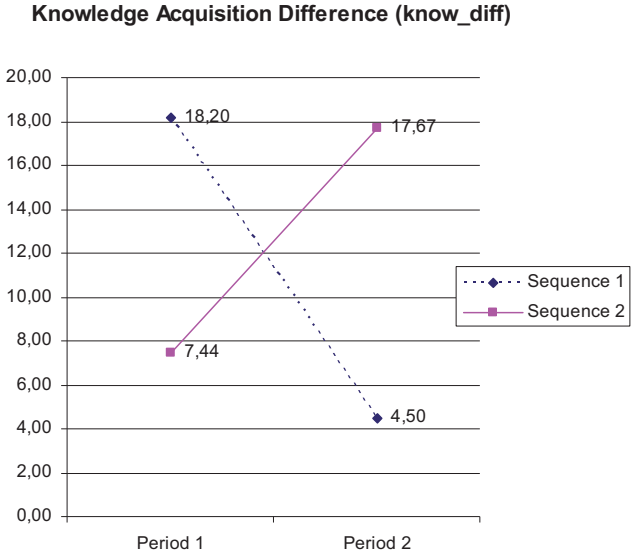


Figure 105 Plot for period effect with respect to sequence

Table 110 Two-tailed independent sample t-test for testing of period effects

	Levene's test for equality of variances		Test for period effect (π)				
	F	p-value	t	df	p-value	Mean Difference	Std. Error Difference
ucorr	4.506	.052	-2.604	16	***.019***	-6.85	2.63
know_diff	.146	.707	1.035	17	.315	3.48	3.36
know_diff_remember	1.635	.222	.730	14	.478	1.37	1.88
know_diff_understand	.001	.971	-1.311	17	.207	-2.48	1.89
know_diff_apply	1.597	.224	3.132	16	***.006***	8.31	2.65
know_diff_analyze	1.710	.208	1.789	17	.091	2.68	1.50
know_diff_create	2.215	.156	.387	16	.704	.22	.57
aeff	.294	.595	1.54	16	.140	.13	.08
acomp	1.597	.224	3.132	16	***.006***	.35	.11
aaccu	2.054	.170	-1.465	17	.161	-.21	.14

Period effects were detected

The significance of Levene's test is above 0.05 for all variables, which suggests that the equal variances assumption is not violated and that the t-test can be performed. Table 110 shows that for most of the variables, the p-values were higher than 2.5%, which means that the null hypotheses of equal period effects ($\pi_1 = \pi_2$) for both sequences could not be rejected. This means that we could expect similar period effects for both periods. A power analysis was conducted to confirm the acceptance of the alternative hypothesis ($\pi_1 \neq \pi_2$). The power for ucorr was 0.652 (df=16; Tcrit=2.119). The power for know_diff_apply was 0.836 (df=16; Tcrit=2.119). The power for acomp was 0.837, (df=16; Tcrit=2.119). This means that we can accept the alternative hypotheses of different period effects for the variables know_diff_apply and acomp. The alternative hypothesis for ucorr cannot be accepted because the

power is lower than 0.80. Therefore, the decision was made to correct for period effects in later analysis steps for the two variables knowledge acquisition difference apply (know_diff_apply) and application completeness (acomp).

It was detected that the period differences for ucorr, know_diff_understand, and know_diff_create were not normally distributed (see Appendix C.4.3). Therefore, a Mann-Whitney-U test was done, which confirms the results of the t-test:

Table 111 Mann-Whitney U test for testing for period effects

	Test for period effect (ucorr)	Test for period effect (level_understand)	Test for period effect (level_create)
Mann-Whitney U	15.500	29.000	39.000
Wilcoxon W	70.500	84.000	84.000
Z	-2.186	-1.317	-.139
Asymp. Sig. (2-tailed)	***.029***	.188	.889
N (LSEP – EP)	10	10	9
N (EP – LSEP)	8	9	9

C.6. Testing the Assumptions for ANCOVA

The homogeneity of variance was tested using Box's M test – the homogeneity of variance could be confirmed for all variables but two ($p = .042$ and $p = .049$; however, slight deviations can be accepted). A repeated measure ANOVA adds another assumption – sphericity (i.e., homogeneity of treatment difference variances). In this experiment, the assumption of sphericity is not relevant because the treatment has only two levels. The analysis of covariance (ANCOVA) adds two further assumptions: linear regression and homogeneity of the regression coefficients, which are more difficult to check. The assumption of linear regression states that the deviations from the regression equation across different treatment levels have a normal distribution with means of zero and that homoscedasticity (i.e., homogeneity of error variances) is fulfilled. The risk of not checking these assumptions (i.e., the linearity of the regression function is not given) is that the regression coefficient b_i does not strive towards the real parameters β_i when the sample size increases (Backhaus, Erichson, Plinke, & Weiber, 2005). Hence, the ANCOVA model assumes that the slopes of the regression lines are the same for each group. That is, the slopes should be parallel. The slope of the regression line is the amount of change in a dependent variable for a given change in a covariate variable. The following table provides the standardized parameters (which makes it easier to compare the parameters amongst the different variables) of the linear equation for each combination of depend-

ent variable and disturbing factor. Y corresponds to the dependent variable, X is the disturbing factor, b_0 corresponds to the constant (i.e., intercept when b_1 is zero), and b_1 to the regression coeffcient (i.e., slope): $Y= b_0 + b_1 * X$.

Table 112 Coefficients of the estimated regression models

		exp_dev	exp_jp	exp_ref	exp_sqa	exp_main	tn	pre-test	inf_qa_LSEP	inf_qa_EP
ucorr LSEP	b_0	35.09	27.90	34.56	35.8	33.73	38.89	25.27	34.56	42.31
	b_1	5.58	2.31	1.57	.71	1.55	-5.00	.59**	.64	-1.20
	Beta	.28	.44	.33	0.18	.41	-.19	.61	.08	-.18
ucorr EP	b_0	27.77	13.4	25.95	25.31	24.96	35.58	18.22	26.62	29.47
	b_1	9.35	4.41**	3.08**	2.43*	2.80**	-15.68*	.68**	.97	.67
	Beta	.44	.79	.61	.57	.70	-.56	.70	.10	.11
know_diff LSEP	b_0	18.98	24.05	18.54	20.00	19.06	16.33	23.07	-2.07	19.93
	b_1	-1.30	-1.32	-.10	-.57	-.27	10.00	-.26	3.91*	-.45
	Beta	-.064	-.25	-.02	-.14	-.07	-.37	-.30	.46	-.08
know_diff EP	b_0	7.88	5.36	4.65	6.90	9.69	6.15	6.64	-5.61	13.42
	b_1	-4.09	.11	.56	-.36	-1.44*	-1.60	-.04	2.21	-1.82
	Beta	-.27	-.03	-.15	.12	-.50	-.08	-.06	.35	-.41
know_diff_remember LSEP	b_0	4.48	4.36	6.54	5.70	3.88	6.05	8.64	-7.34	3.29
	b_1	2.60	.33	-.36	.03	.71	-1.36	-.14	2.51*	.56
	Beta	.21	.10	-12	.01	.31	-.08	-.29	.50	.17
know_diff_remember EP	b_0	2.17	2.61	1.80	1.00	2.02	2.08	.89	.23	1.37
	b_1	-.45	-.15	.07	.32	-.03	-.68	.05	.31	.13
	Beta	-.10	-.13	-07	.36	-.04	-.12	.26	.16	.11
know_diff_understand LSEP	b_0	7.24	7.91	6.13	8.27	5.82	5.48	6.35	-2.15	7.75
	b_1	-2.14	-.40	.02	-.72	.13	3.41	-.01	1.59	-.37
	Beta	-.20	-.14	.01	-.32	.06	.24	-.03	-.35	.12
know_diff_understand EP	b_0	2.50	2.25	2.14	1.76	3.28	1.72	3.45	-2.39	2.21
	b_1	-2.01	-.17	-.30	-.09	-.68	-1.14	-.09	.774	-.16
	Beta	-.25	-.08	-.15	-.05	-.43	-.10	-.27	.224	-.07
know_diff_apply LSEP	b_0	11.34	3.41	9.83	10.05	12.94	15.54	12.75	9.65	8.29
	b_1	4.23	2.32*	1.72	1.16	.19	-10.40	.01	.73	1.07
	Beta	.23	.47	.38	.30	.05	-.42	.02	.09	.19
know_diff_apply EP	b_0	10.05	6.33	8.31	11.64	10.57	8.74	7.15	10.51	14.34
	b_1	-2.11	.62	.32	-.90	-.59	1.30	.08	-.32	-1.21
	Beta	-.15	.16	.09	-.30	-.21	.07	.14	-0.5	-.30

		exp_dev	exp_jp	exp_ref	exp_sqa	exp_main	tn	pre-test	inf_qa_LSEP	inf_qa_EP
know_diff_analyze LSEP	b_0	5.62	9.06	4.67	4.73	7.18	3.44	6.43	7.29	7.57
	b_1	-1.36	-.95	.13	.07	-.83	7.50*	-.09	-.45	-.65
	Beta	-.13	-.34	.05	.03	-.41	.53	-.19	-.10	.21
know_diff_analyze EP	b_0	2.26	.30	1.30	3.52	3.30	1.82	1.96	-3.24	8.25
	b_1	-.52	.40	.33	-.52	-.49	0.90	.00	1.02	-1.42**
	Beta	-.06	.18	.16	-.30	-.30	.08	.01	.29	-.60
know_diff_create LSEP	b_0	1.55	1.23	.68	1.24	1.372	1.22	.57	.69	2.10
	b_1	-.65	0.00	.26	-.01	-.06	0.00	.03	.10	.21
	Beta	-.19	0.00	.32	-.01	-.09	0.00	.21	.07	.22
know_diff_create EP	b_0	1.03	.36	.261	.519	.99	.581	.44	-.96	1.11
	b_1	-1.17	.02	.33	-.03	-.20	-.68	.00	.26	-.15
	Beta	-.36	.02	.42	-.03	-32	-.16	-.01	.20	-.17
aeff LSEP	b_0	.58	.11	.35	.51	.49	.55	.59	.55	.36
	b_1	-.22	.08	.06	-.01	-.01	-.39	-.01	-.02	.02
	Beta	-.21	.30	.21	-.05	-.03	-27	-.16	-.03	.07
aeff EP	b_0	.33	.18	.28	.39	.30	.26	.26	.21	.36
	b_1	-.14	.02	0.00	-.04*	-.01	.05	.00	.01	-.02
	Beta	-.33	.20	-.06	-.49	-19	.08	.00	.05	-.19
acomp LSEP	b_0	.47	.14	.410	.42	.54	.65	.53	.40	.346
	b_1	.17	.10*	.07	.05	.01	-.43	.00	.03	.04
	Beta	.22	.47	.38	.30	.05	-.42	.02	.09	.19
acomp EP	b_0	.42	.26	.35	.48	.44	.36	.30	.44	0.60
	b_1	-.09	.02	.01	-.04	-.02	.05	.00	-.01	-.05
	Beta	-.15	.16	.09	-.31	-.21	.07	.14	-.05	-.30
aaccu LSEP	b_0	.77	.63	.63	.71	.81	.74	.70	1.09	.71
	b_1	-.11	.02	.04	.00	-.03	-.13	.00	-.07	.00
	Beta	-.16	.11	.26	.01	-.27	-.14	.06	-.26	.02
aaccu EP	b_0	.76	.89	.74	.63	.73	.52	.88	.03	.32
	b_1	-.37*	-.07	-.08*	-.02	-.06	.27	-.02*	.10	.06
	Beta	-.55	-.40	-.48	-.12	-.44	.30	-.56	.037	.31

Significant correlations ($p<0.01$ have been marked with a **; $p<0.05$ with *)

In order to interpret the regression parameters, Pearson's correlation coefficients between each dependent variable and disturbing factor are displayed in Table 113. Significant correlations p<0.01 have been marked with ** and with * for p<0.05.

Table 113 Pearson's correlations between the dependent variables and disturbing factors

Correlation	exp_dev	exp_jp	exp_ref	exp_sqa	exp_main	tn	Pre-test	inf_qua_LSEP	inf_qua_EP
ucorr LSEP	.283	.444	.330	.177	.411	-.161	.611**	.140	-.182
ucorr EP	.444	.795**	.607**	.567*	.697**	.115	.697**	.099	.060
know_diff LSEP	.042	-.098	.012	-.057	-.020	-.077	-.302	.480*	-.077
know_diff EP	-.273	-.001	.145	-.129	-.501*	-.400	-.061	.353	-.400
know_diff _remember LSEP	.320	.075	-.163	-.026	.527*	.302	-.292	.492*	.168
know_diff _remember EP	.045	.550*	.401	.191	.171	-.228	.256	.163	.073
know_diff _understand LSEP	-.152	-.090	.016	-.289	.076	-.123	-.028	.358	-.123
know_diff _understand EP	-.309	-.172	-.175	-.112	-.451	-.048	-.268	.224	-.048
know_diff_apply LSEP	.238	.538*	.268	.242	.171	.244	.016	.133	.193
know_diff_apply EP	-.201	.167	-.052	-.458	-.155	-.318	.143	-.053	-.311
know_diff_analyze LSEP	.013	-.140	.085	.119	-.321	.559**	-.191	-.047	-.208
know_diff_analyze EP	-.091	.119	.150	-.314	-.313	.060	.013	.290	-.591**
know_diff_create LSEP	-.004	.261	.332	.110	.084	-.206	.209	.108	-.222
know_diff_create EP	-.181	.431	.495*	.104	-.079	-.153	-.006	.196	-.174
aeff LSEP	-.113	.252	.280	.078	-.073	.008	-.156	-.005	.074
aeff EP	-.200	.300	-.026	-.383	-.132	-.208	.003	.048	-.209
acomp LSEP	.238	.538*	.268	.242	.171	.244	.016	.133	.193
acomp EP	-.201	.167	-.052	-.458	-.155	-.318	.143	-.053	-.311
aaccu LSEP	-.218	.023	.230	-.041	-.292	.021	.055	-.277	.021
aaccu EP	-.459*	-.305	-.459*	-.081	-.408	.292	-.559*	.369	.292

High correlations correspond to a linear regression and the chance to reduce error variance

There is a relationship between the correlations and the regression parameters: Significant regression parameters were found for most dependent variable/disturbing factor combinations where Pearson's correlation coefficient was significantly high. Hence, high correlations correspond to linear regression, whereas lower correlations refer to non-linear correlations or no correlation at all. Further, the more the disturbing factor correlates with the dependent variables, the more the error vari-

ance is reduced by the disturbing variable. When the correlation is not significant, the reduction of the error variance is due to chance. Therefore, the table gives first indications about which disturbing factors may reduce error variance. All the cells with a significant regression coefficient (i.e., slope) have a *beta* higher than 0.46. Table 112 and Table 113 provide guidance for further applications of ANCOVA and should help to prevent looking for a needle in a haystack and unsystematic poking into the data.

Looking at the data of Table 112, the disturbing factors exp_jp and pretest correlate with several dependent variables. It can be seen, for example, that the understanding correctness for EP depends much more on human experience than the understanding correctness for LSEP. Almost all experience disturbing factors correlate significantly with the understanding correctness (ucorr). This makes the effect of learning spaces on understanding correctness less dependent on human experience. It can be assumed that the p-value for hypothesis tests will decrease further when ANCOVA is applied with the experience factors for ucorr compared to the results of the corresponding t-test, respectively Wilcoxon test.

Check for homoscedasticity

Regarding homoscedasticity, this check was done graphically by looking at a scatter plot of the residuals. The x-axis stands for the estimated values of the dependent variable ($\hat{y}$) based on the regression equation. The y-axis stands for the standardized residuals of the observed values. Figure 106 shows one example of ucorr (EP). The assumption of homoscedasticity was fulfilled for all variables because no obvious relationship between $\hat{y}$ and the residuals could be found.

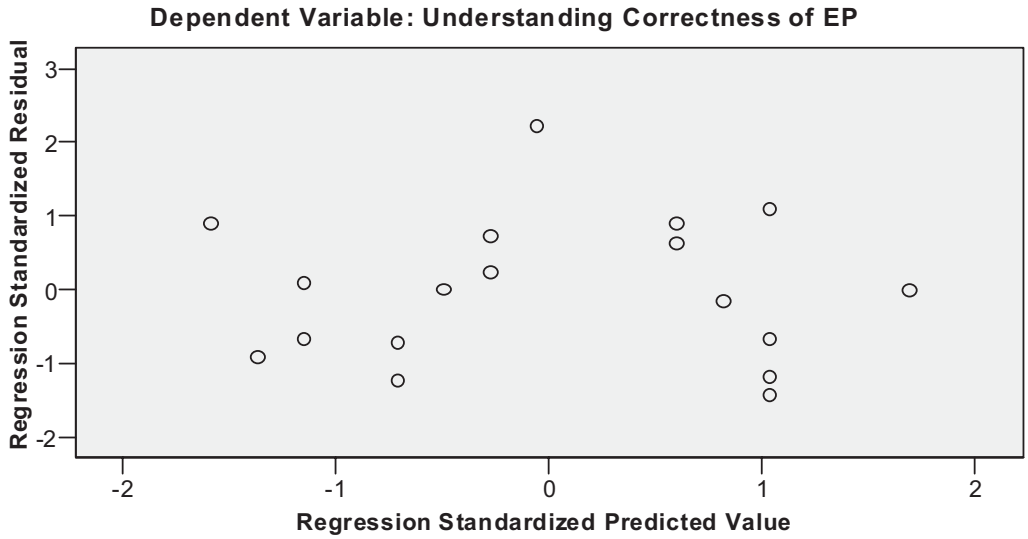


Figure 106 Scatter plot for testing of homoscedasticity

Check for equality of slopes of regression lines

Furthermore, the equality of slopes for both treatments has to be investigated. This can be done either graphically or by performing a F-test and testing whether there is a significant treatment*disturbing variable interaction. This interaction should be nonsignificant. If the interaction is significant (p-value < 0.05), then the slopes for the two treatments are significantly different and the ANCOVA assumption has been violated. Looking at the values in Table 112, it can be seen that several regression coefficients differ a lot between both treatments. The regression parameters were calculated based on data from the LSEP treatment or EP treatment separately. ANCOVA will use both data sets and will create a so-called pooled slope (i.e., a kind of average slope of both treatments) to reduce the difference between the slopes. Therefore, large differences in Table 112 do not mean that these covariates are not suitable for ANCOVA. Table 114 shows the results of the univariate ANCOVA, whose intent was to uncover different slopes of regression lines (i.e., these disturbing factors should not be used in the ANCOVA):

Table 114 P-values of treatment *disturbing factor covariate

Disturbing variable / Dep. variable	exp_dev	exp_jp	exp_ref	exp_sqa	exp_main	tn	pre-test	inf_qua_LSEP	inf_qua_EP
ucorr	.577	.155	.327	.204	.275	.224	.621	.922	.429
know_diff	.423	.747	.757	.905	.239	.103	.381	.409	.469
know_diff_remember	.259	.735	.307	.764	.059	.859	.527	.178	.175
know_diff_understand	.777	.891	.625	.499	.160	.234	.560	.525	.780
know_diff_apply	.211	.153	.322	.055*	.359	.185	.719	.565	.118
know_diff_analyze	.791	.450	.930	.240	.765	.066	.507	.381	.406
know_diff_create	.666	.801	.852	.927	.647	.413	.705	.630	.782
aeff	.958	.836	.312	.272	.969	.547	.877	.896	.635
acomp	.211	.153	.322	.055*	.359	.185	.719	.565	.118
aaccu	.450	.331	.038*	.905	.711	.147	.058*	.054*	.421

The cells marked with * were checked graphically by a scatter plot (dep. variable*disturbing variable).

Lebenslauf

Name	Eric Ras	
Wohnort	Carl-Euler-Str. 53 67663 Kaiserslautern	
Geboren	24.07.1975	
Geburtsort	Luxemburg	
Familienstand	Verheiratet (ein Kind)	
Staatsangehörigkeit	Luxemburgisch	
Werdegang	1981-1987	École primaire Wincrange
	1987-1994	Lycée Technique Wiltz (Abitur)
	1994-2000	Technische Universität Kaiserslautern (Dipl.-Technoinform.)
	2000-heute	Wissenschaftlicher Mitar- beiter am Fraunhofer Institut Experimentelles Software Engineering, Kaiserslautern

Kaiserslautern, den 18. Dezember 2008

PhD Theses in Experimental Software Engineering

Volume 1 **Oliver Laitenberger** (2000), *Cost-Effective Detection of Software Defects Through Perspective-based Inspections*

Volume 2 **Christian Bunse** (2000), *Pattern-Based Refinement and Translation of Object-Oriented Models to Code*

Volume 3 **Andreas Birk** (2000), *A Knowledge Management Infrastructure for Systematic Improvement in Software Engineering*

Volume 4 **Carsten Tautz** (2000), *Customizing Software Engineering Experience Management Systems to Organizational Needs*

Volume 5 **Erik Kamsties** (2001), *Surfacing Ambiguity in Natural Language Requirements*

Volume 6 **Christiane Differding** (2001), *Adaptive Measurement Plans for Software Development*

Volume 7 **Isabella Wieczorek** (2001), *Improved Software Cost Estimation A Robust and Interpretable Modeling Method and a Comprehensive Empirical Investigation*

Volume 8 **Dietmar Pfahl** (2001), *An Integrated Approach to Simulation-Based Learning in Support of Strategic and Project Management in Software Organisations*

Volume 9 **Antje von Knethen** (2001), *Change-Oriented Requirements Traceability Support for Evolution of Embedded Systems*

Volume 10 **Jürgen Münch** (2001), *Muster-basierte Erstellung von Software-Projektplänen*

Volume 11 **Dirk Muthig** (2002), *A Light-weight Approach Facilitating an Evolutionary Transition Towards Software Product Lines*

Volume 12 **Klaus Schmid** (2003), *Planning Software Reuse – A Disciplined Scoping Approach for Software Product Lines*

Volume 13 **Jörg Zettel** (2003), *Anpassbare Methodenassistenz in CASE-Werkzeugen*

Volume 14 **Ulrike Becker-Kornstaedt** (2004), *Prospect: a Method for Systematic Elicitation of Software Processes*

Volume 15 **Joachim Bayer** (2004), *View-Based Software Documentation*

Volume 16 **Markus Nick** (2005), *Experience Maintenance through Closed-Loop Feedback*

Volume 17 **Jean-François Girard** (2005), *ADORE-AR: Software Architecture Reconstruction with Partitioning and Clustering*

Volume 18 **Ramin Tavakoli Kolagari** (2006), *Requirements Engineering für Software-Produktlinien eingebetteter, technischer Systeme*

Volume 19 **Dirk Hamann** (2006), *Towards an Integrated Approach for Software Process Improvement: Combining Software Process Assessment and Software Process Modeling*

Volume 20 **Bernd Freimut** (2006), *MAGIC: A Hybrid Modeling Approach for Optimizing Inspection Cost-Effectiveness*

Volume 21 **Mark Müller** (2006), *Analyzing Software Quality Assurance Strategies through Simulation. Development and Empirical Validation of a Simulation Model in an Industrial Software Product Line Organization*

Volume 22 **Holger Diekmann** (2008), *Software Resource Consumption Engineering for Mass Produced Embedded System Families*

Volume 23 **Adam Trendowicz** (2008), *Software Effort Estimation with Well-Founded Causal Models*

Volume 24 **Jens Heidrich** (2008), *Goal-oriented Quantitative Software Project Control*

Volume 25 **Alexis Ocampo** (2008), *The REMIS Approach to Rationale-based Support for Process Model Evolution*

Volume 26 **Marcus Trapp** (2008), *Generating User Interfaces for Ambient Intelligence Systems; Introducing Client Types as Adaptation Factor*

Volume 27 **Christian Denger** (2009), *SafeSpection – A Framework for Systematization and Customization of Software Hazard Identification by Applying Inspection Concepts*

Volume 28 **Andreas Jedlitschka** (2009), *An Empirical Model of Software Managers' Information Needs for Software Engineering Technology Selection A Framework to Support Experimentally-based Software Engineering Technology Selection*

Volume 29 **Eric Ras** (2009), *Learning Spaces: Automatic Context-Aware Enrichment of Software Engineering Experience*

Software Engineering has become one of the major foci of Computer Science research in Kaiserslautern, Germany. Both the University of Kaiserslautern's Computer Science Department and the Fraunhofer Institute for Experimental Software Engineering (IESE) conduct research that subscribes to the development of complex software applications based on engineering principles. This requires system and process models for managing complexity, methods and techniques for ensuring product and process quality, and scalable formal methods for modeling and simulating system behavior. To understand the potential and limitations of these technologies, experiments need to be conducted for quantitative and qualitative evaluation and improvement. This line of software engineering research, which is based on the experimental scientific paradigm, is referred to as 'Experimental Software Engineering'.
In this series, we publish PhD theses from the Fraunhofer Institute for Experimental Software Engineering (IESE) and from the Software Engineering Research Groups of the Computer Science Department at the University of Kaiserslautern. PhD theses that originate elsewhere can be included, if accepted by the Editorial Board.

ISBN: 978-3-8396-0016-0

Fraunhofer IESE

Technische Universität Kaiserslautern
AG Software Engineering

PhD Theses in Experimental Software Engineering