Workplace Learning in Reuse-Oriented Software Engineering

Eric Ras
(Fraunhofer Institute for Experimental Software Engineering, Germany
eric.ras@iese.fraunhofer.de)

Jörg Rech
(Fraunhofer Institute for Experimental Software Engineering, Germany
joerg.rech@iese.fraunhofer.de)

Björn Decker
(Fraunhofer Institute for Experimental Software Engineering, Germany
bjoern.decker@iese.fraunhofer.de)

Abstract: Today, *reuse-oriented software engineering* covers the process of the development and evolution of software systems by reusing existing experience (i.e., products, processes, and knowledge). One of the major problems of software reuse is the lack of knowledge and skills for understanding reusable experience. This paper explains how the reuse process can be used to support individual learning on the one hand, and how learning can improve the selection of reuse experience and their application on the other hand. The paper emphasizes the importance of context in the domain of reuse and how context information can be used to compose so-called Learning Spaces from Learning Components. Learning Spaces didactically enrich reusable experience and enhance experiential learning. The approach uses Wikis as a base technology for presenting and structuring learning content.

Keywords: software reuse, software engineering, workplace learning, experiential learning, wiki, learning component, learning space
Categories: D.2, D.2.13, H.2.8, H.3, H.4, H.5, K.3

1 Introduction

The reuse of existing knowledge and experience is a fundamental practice in many sciences. Engineers often use existing components and apply established processes to construct complex systems. Without the reuse of well-proven experience (i.e., software products, processes, and knowledge), engineers have to rebuild and relearn this experience again and again. Following this path, large amounts of documented experience has been amassed in the experience management systems of software engineering organizations. These experience management systems are targeted to the *organizational perspective* in order to improve the sharing of experience among members of an organization by 1) encouraging the individuals to make their experience explicit by creating knowledge chunks that can be stored in experience bases for later reuse or 2) participating in communities of practice. They are intended to support short-term problem-solving as well as long-term strategic learning. Contrary to that, e-Learning systems emphasize an *individual perspective*, as they

focus on the individual acquisition of new knowledge (i.e., individual competence development) and the socio-technical means to support this internalization process.

Systems used to support reuse-oriented software engineering belong to the category of knowledge management systems, since they support the exchange of knowledge between members of an organization. During the last two decades, reuse-based approaches have been evaluated from many different perspectives and many aspects for failure and success of such systems have been identified. Karlsson identifies three main reasons for failure: *Search (and Retrieval)*, *Evaluation*, and, *Adaptation*. 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 required to understand and adapt the artifact in contrast to the effort for its initial creation [Karlsson 95]. Factors that have been identified, but not addressed explicitly by current approaches so far, are the lack of understanding of reuse content, awareness of available content, and the quality of the content regarding learning efficiency and effectiveness.

This paper aims at first investigating the first problem related to understanding during reuse and second, at providing an approach on how experiential learning at the workplace can be improved by integrating learning into the reuse process and by explicitly considering the context of the software engineer for the creation of so-called Learning Spaces. [Section 2] gives an introduction to reuse in software engineering. [Section 3] states the problems related to understanding on experience reuse. [Section 4] motivates the importance of context in reuse. [Section 5] explains the approach. [Section 6] provides the current state of implementation and an outlook.

2 Reuse in Software Engineering

Today, *reuse-oriented software engineering* covers the process of development and evolution of software systems by reusing existing experience. Most approaches focus mainly on reuse of software products (e.g., source code) in order to develop complex software systems in shorter periods of time or with a higher quality by reusing proven, verified, and tested components from internal or external sources. However, beside the reuse of source code, systematic reuse should also support reuse of all software artifacts such as [Basili, Rombach 91]: software products (requirements, designs, measurement plans, etc.), software processes (lifecycle models, process models, methods, etc.), and further knowledge about software (forms and templates, quality and defect models, lessons learned, best practices, and observations). Since the 1980s, the systematic reuse and management of products, processes, and knowledge was refined and named *Experience Factory* (EF) [Basili et al. 94]. In this field, also known as *Experience Management*, research methods and techniques for the management, elicitation, and adaptation of reusable experience from SE projects have been developed.

Beside Karlsson's reuse process, many other reuse processes have been defined and applied in the past, depending on their artifacts to be reused and the context of individuals and organization. Prieto-Diaz and Freeman state that three steps for code reuse are necessary: a) accessing, b) understanding, and c) adapting the code [Prieto-

Diaz Freeman 87]. Biggerstaff and Richter list four steps for effective reuse: a) find, b) understand, c) modify, and d) compose artifacts [Biggerstaff, Richter 89]. Basili and Rombach define the reuse process by requiring mechanisms for: a) identifying, b) evaluating, c) selecting, and d) modifying artifacts from the experience base [Basili, Rombach 91]. Common to all these models of the reuse process is the essential activity of understanding reuse artifacts. The next section lists reuse problems that are related to the aspect of understanding.

3 Understanding of Experience in Software Reuse

Experience management systems as used in software engineering focus mainly on acquisition, storage, and retrieval, and less on the learning processes themselves or the personal learning needs of the software developers. More specific problems related to the understanding of experience (i.e., products, processes, and knowledge) are listed below:

- *Problem of describing experience*: The quality of the reported and recorded experience highly depends on the individual skills of the contributor, e.g., the ability to structure and formulate the content with accuracy, and to describe it properly according to the needs of the target audience. Without assistance, this leads to quality discrepancies, gaps in the experience base, and problems in understanding.
- *Problem of understanding and applying documented experience*: Experience is often documented by domain experts. Expert knowledge is somehow ‘routine’. This makes it challenging for experts to document experiences appropriately and to make them understandable for others. Novices lack software engineering background knowledge and are not able to connect the experience to their experience base. Hence, they often misinterpret or even do not understand other people’s documented experience. A more detailed summary of problems related to understanding and learning from documented experience can be found in [Ras, Weibelzahl 04] and [Frakes et al. 91].
- *Problem of technical support*: Most systems focus on the storage and retrieval of information. The search is mostly initiated by the user and current information about the context is not included automatically in the search statement. The systems do not didactically structure the content in order to improve learning efficiency and experiential learning, in particular.

Many researchers have applied concepts from the cognitive psychology field to software reuse understanding. Dusink studies the application of problem solution methods in reuse problems, such as understanding what a reusable component does and when to apply it [Dusink 91]. Fischer et al. present cognitive tools to support the retrieval and comprehension of reusable components and talk about 'reflection-in-action' as the mode in which a knowledge worker can reach previously tacit knowledge [Fischer et al. 91]. Wieser et al. present cognitive experience transfer tasks, which focus more on the human-oriented point of view and less on the organization or activities aspects (i.e., activating experience, collecting experience, processing and storing experience, making experience available for the task at hand) [Wieser et al. 99]. According to Kolb, individuals most effectively learn from experiences when all four phases of Kolb’s *Experiential Learning Circle* are covered:

concrete experience, observation and reflection, forming abstract concepts, and testing in new situations [Kolb 84]. By documenting an experience for later reuse (i.e., forming abstract concepts), a software engineer profits from his or her own perception that leads to the experience, his or her own observation, and the reflection about it. When a software engineer other than the author wants to reuse a documented experience, he or she will lack contextual information, observational qualities, and reflective attributes deriving from it. Hence, systems should focus on the delivery of appropriate content in addition to the reusable experience in order to support knowledge construction as described in Kolb's learning cycle. The next section explains how context information can help to address the problems elaborated earlier in this section by creating learning content that enhances experiential learning.

4 The Role of Context in Software Reuse

The context of a reusable artifact can be seen as all information that is not part of the artifact or that describes the artifact itself and is relevant to this artifact's use. Context explains the environment the experience was created in and is a way of giving experience meaning and focus. The more understandable and focused it is, the more effectively it can be captured and reused in a given situation.

The importance of context information in software reuse has been highlighted very often. Basili and Rombach consider context information, besides object and interface information, as essential in their reuse model [Basili, Rombach 91]. Tautz says that the characterization of a reuse object must include context information [Tautz 00]. In addition to these approaches, much work has been done on context description for specific types of experience. 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 et al. 99]. So far, context information in software reuse has been mostly used for classification and retrieval purposes. Other models emphasize more on the collaborative characteristic of software development and focus on the description of the group context. Rosa et al. consider 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 et al. 03]. 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. 04]. However, they do not provide any details about those facets.

The context vector proposed in this work is based on the previously mentioned work and has to fulfill the following requirements:

- gives the artifact meaning and focus,
- describes the situation in which the artifact is relevant and where it has been created (i.e., its applicability),
- relates the artifact to product, process, and project,
- supports the description of individual context and group context,
- enables the integration of learning into the working process and finally,

- provides input parameters for the creation of Learning Spaces for enhancing experiential learning.

This results in the following context categories with example refinements that can be used to describe an experience:

- individual (e.g., role, skill and competence profile, learning preference, activity history),
- group (e.g., team size, team members, team skills and competencies),
- product (e.g., type of product, complexity, quality, application context),
- process (e.g., activity, lifecycle model),
- project (e.g., size, effort, resources, costs),
- organization (e.g., competence development strategies, corporate quality strategy, business targets) and finally,
- customer (e.g., contact, contract, business domain).

The refinement of each context category depends on the domain where reuse is applied. In addition to the context characterization, each artifact needs a description vector of the artifact itself: classification, abstract, keywords, definition, explanation, relation to other artifacts, etc. Each characteristic is an attribute/value pair where the value is consistent with a predefined vocabulary. The next section shows how Learning Spaces are created.

5 Creating Learning Space based on Context Information

The aim of the approach presented in this section is to enhance experience understanding by supporting experiential learning. The approach uses Wikis as a base technology for presenting and structuring learning content, and composes so-called Learning Spaces from Learning Components. This section explains the main concepts used, the reuse process with integrated learning activities, and the creation of so-called Learning Spaces based on context information.

Wikis have shown that they are a good facility for easy content creation and simple knowledge management in agile software engineering [Decker et al. 06]. Learning content is aggregated into a so-called Learning Space, i.e., a network of Wiki pages that consist of Learning Components and Learning Elements (see [Fig. 1]): *Learning Elements* are the most basic learning resources. They are electronic representations of media, such as images, text, sound, or any other piece of data that can serve as a learning resource when aggregated with other Learning Elements to form a Learning Component. *Learning Components* are units of instruction that contain at least one Learning Element. A Learning Component represents the lowest granularity of a learning resource that can be reused by the system for learning purposes. The difference between a Learning Component and a Learning Element is that a Learning Component is related to a learning activity and a learning objective, while Learning Elements are the building blocks of Learning Components. In addition, they can be referenced as a learning resource by the system (e.g., by using hyperlinks). Learning Elements can be compared to SCORM assets and Learning Components are similar to SCORM Sharable Content Objects. The reason why we use a different terminology is that the term *element* reflects more that it is part of a whole and the term component comes from component-based software engineering (see [Memmel et al. 06] for more details). A *Learning Space* follows a specific global

learning goal and is created based on context information of the current situation. The goal of a Learning Space is to provide a learning environment for experiential learning at the workplace. The composition of a Learning Space follows the experiential learning method by using an instructional design framework, and by generating a logical learning activity tree. More details about the composition process can be found in [Rech et al. 06].

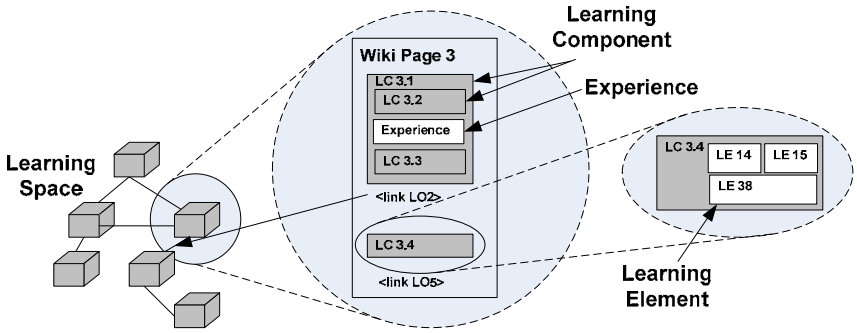

Figure 1: Learning Space composition

[Fig. 2] shows the software product reuse process and the different context categories. The grey area illustrates the scope of the Learning Space. A Learning Space intends to improve the developer's understanding during the evaluation, resp. selection of the retrieved artifact and during the modification of the artifact, if this is required. The evaluation activity is one of the most time consuming activities and is basically cognitive. The modification of the artifact requires that the developer fully understand the artifact and its context (i.e., context description of its creation) and the commonalities with, respectively differences from, the current context.

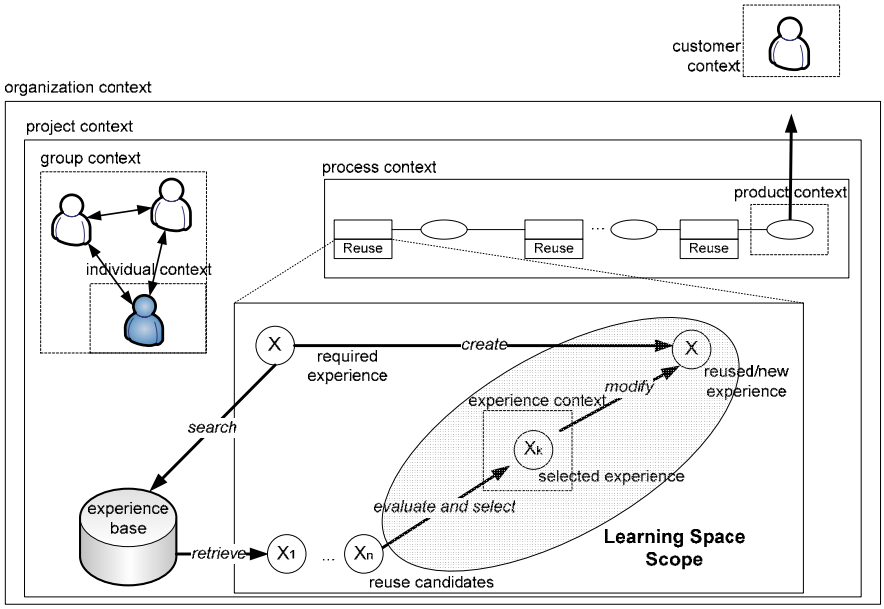

Figure 2: Reuse process and integration of learning

[Fig. 1] illustrates how an experience is integrated into a Learning Component. Learning Components relevant to the current context help the engineer to make concrete experience and new observations, to reflect, to form abstract concepts on his or her own, and to test these abstractions in the current situation, and hence to better understand the experience to be reused.

The Learning Space is generated based on a previously calculated context gap. This gap is calculated in four different steps: The first step is done during retrieval of reuse candidates. Similarity-based retrieval calculates the distance between the current software development context (i.e., project, process, product, etc.) and the artifact context based on weighted attributes and distance measures. The second step derives from the context of the retrieved artifacts: Which skills and competencies are necessary for reusing it (i.e., based on concepts used from a predefined competence ontology)? The third step compares and reduces the derived topic list depending on the individual profile of the user (i.e., skill and competence, task history, etc.) and the organizational context (e.g., prioritizing according to corporate competence development strategies). The last step generates a Learning Space by checking the group context in order to decide whether learning components contain learning content or simply refer to knowledgeable colleagues. The individual learning preferences are used to select a suitable experiential learning variant and to sequence and present the content by means of composed Wiki pages (see [Rech et al. 06] for the details about how to realize learning methods).

6 Current State of Implementation and Future Work

We have described characteristics of context information required to enable workplace learning in software engineering and have shown how learning can be integrated into the reuse process. Several evaluations have been done regarding user acceptance and the effectiveness of Wiki usage in agile Software Engineering. However, no proof can be given for the success of the Learning Space approach yet. The technical problems of extending the Wiki with e-learning functionalities have been solved and a basic set of learning content is available in the system. We expect first evaluation results of the approach by the end of this year by conducting a controlled experiment with software engineering students. The focus will lie on the impact of the learning method on reuse efficiency. Beside the implementation of the experiential learning method, another issue is to capture context information automatically during software development and to calculate context gaps between context vectors of the experience to be reused and the actual context more accurately.

References

[Araujo et al. 04] Araujo, R., Santoro, F. M., Brézillon, P., Borges, M. R., Rosa, M. G. P.: "Context Models for Managing Collaborative Software Development Knowledge"; Proc. First International Workshop on Modeling and Retrieval of Context (KI 2004), MRC2004, Ulm, Germany (2004).

[Basili, Rombach 91] Basili, V. R., Rombach, H. D.: "Support for Comprehensive Reuse"; Software Engineering Journal, 6, 5 (1991), 303-316.

[Basili et al. 94] Basili, V. R., Caldiera, G., Rombach, H. D.: "Experience Factory"; J. J. Marciniak, John Wiley & Sons, New York, (1994), 469-476.

[Biggerstaff, Richter 89] Biggerstaff, T. J., Richter, C.: "Reusability Framework, Assessment, and Directions"; ACM Press, (1989), 1-17.

[Damiani et al. 99] Damiani, E., Fugini, G., Bellettini, C.: "A hierarchy-aware approach to faceted classification of objected-oriented components"; ACM Trans. Softw. Eng. Methodol., 8, 3 (1999), 215-262.

[Decker et al. 06] Decker, B., Rech, J., Ras, E., Klein, B., Hoecht, C.: "Using Wikis to Manage Use Cases: Experience and Outlook"; Proc. International Workshop on Learning Software Organizations and Requirements Engineering (LSO+RE 2006), Hannover, Germany (2006),

[Dusink 91] Dusink, E. M.: "Software Engineering and Reuse"; Proc. Fourth International Workshop on Software Reusability, Virginia, USA (1991).

[Fischer et al. 91] Fischer, G., Henninger, S., Redmiles, D.: "Cognitive Tools for Locating and Comprehending Software Objects for Reuse"; Proc. 13th International Conference on Software Engineering, (1991), 318-328.

[Frakes et al. 91] Frakes, W. B., Biggerstaff, T. J., Prieto-Diaz, R., Matsumura, K., Schaefer, W.: "Software reuse: is it delivering?" IEEE Computer Society Press, Austin, Texas, United States, (1991), 52-59.

[Karlsson 95] Karlsson, E. A.: "Software Reuse: A Holistic Approach"; Wiley and Sons, (1995).

[Kolb 84] Kolb, D. A.: "Experiential learning : experience as the source of learning and development"; Prentice-Hall, Englewood Cliffs, N.J., (1984).

[Memmel et al. 06] Memmel, M., Ras, E., Jantke, J. K., Yacci, M.: "Approaches to Learning Object Oriented Instructional Design"; A. Koohang and K. Harman (eds), Informing Science, (in press)

[Prieto-Diaz, Freeman 87] Prieto-Diaz, R., Freeman, P.: "Classifying Software for Reusability"; IEEE Software, 4, 1 (1987).

[Ras, Weibelzahl 04] Ras, E., Weibelzahl, S.: "Embedding Experiences in Micro-didactical Arrangements"; Proc. SEKE 2004, Springer-Verlag, Banff, Canada (2004), 55-66.

[Rech et al. 06] Rech, J., Ras, E., Decker, B.: "Riki: A System for Knowledge Transfer and Reuse in Software Engineering Projects: Strategies beyond Tools"; M. Lytras and A. Naeve (eds), Idea Group, Inc., (in press)

[Rosa et al. 03] Rosa, M. G. P., Borges, M. R. S., Santoro, F. M.: "A Conceptual Framework for Analyzing the Use of Context in Groupware"; Proc. 9th International Workshop on Groupware: Design, Implementation, and Use, Springer Berlin/Heidelberg, Autrans, France (2003), 300-313.

[Tautz 00] Tautz, C.: "Customizing Software Engineering Experience Management Systems and Related Processes for Sharing Software Engineering Experience"; Department of Computer Science, University of Kaiserslautern, Germany, Kaiserslautern (2000).

[Wieser et al. 99] Wieser, E., Houdek, F., Schneider, K.: "Systematic Experience Transfer: Three Case Studies From a Cognitive Point of View"; Proc. International Conference on Product Focused Software Process Improvement (PROFES), Oulu, Finland (1999), 323-344.