INFORMATIONSTECHNOLOGIE UND ÖKONOMIE
Herausgegeben von Christian Becker, Wolfgang Gaul, Armin Heinzl,
Alexander Mädche und Martin Schader 54

Sven Scheibmayr

Graphical User Interface Prototyping for Distributed Requirements Engineering

ANWENDUNGEN
PROBLEME
WISSEN

PL ACADEMIC RESEARCH

1. Introduction

1.1. Problem Outline

The objective of Software Engineering (SE) is to build high quality software within a given time and with a predetermined budget (Sommerville, 2007). Often, though, software development projects still struggle to accomplish these objectives and many fail (Charette, 2005; The Standish Group, 2009). Studies show that in many cases, problems in the early phase of software development lead to cost or time overruns, rework, bad quality, and eventually to the failure of the project (Procaccino et al., 2006; Hofmann and Lehner, 2001).

Requirements Engineering (RE) is the phase of SE which deals with this early phase of software development (Sommerville and Sawyer, 1997; Nuseibeh and Easterbrook, 2000). RE can be defined as follows: "Requirements engineering is the disciplined application of proven principles, methods, tools, and notations to describe a proposed system's intended behavior and its associated constraints"(Hsia et al., 1993). RE comprises activities such as the elicitation of the requirements, their specification, and their management. The main task of RE is essentially the transfer of knowledge between stakeholders in the project and the creation of a common understanding of "what to build". Stakeholders are individuals or organizations which actively participate in a software project or whose interests influence the project (Hofmann and Lehner, 2001).

Today, globalization also affects the processes of how software is built and many development projects nowadays are conducted in globally distributed

environments, where project stakeholders are spread over multiple locations and often work in different time zones (Karolak, 1998). This distribution of stakeholders leads to a variety of challenges in collaboration and communication (Herbsleb and Moitra, 2001) and needs for research in that field (Hargreaves et al., 2004). Stakeholders are not able to communicate in the same way as in face-to-face meetings at one location. Informal and ad-hoc communication is impeded[1].

Synchronous rich communication needs significant effort and tools. Furthermore, multiple time-zones often constrain the available time for synchronous communication. Cultural differences and language barriers make communication and collaboration more difficult. As RE requires extensive communication, these problems make it even more challenging in global and distributed projects. To ensure an efficient and effective communication over distance, solid methods and tools have to be developed and evaluated. The design of these methods and tools should be informed by existing theoretical knowledge about communication and coordination processes in order to ensure the effectiveness of these methods and tools (Hevner et al., 2004).

This thesis focuses on the following general question:

> *"How can it be ensured that requirements engineering leads to high quality requirements in distributed software development projects?"*

The general objective of this research is to support requirements engineering to come up with requirements in such a way that the system to be build meets the stakeholders' needs. The search for these requirements is usually conducted in the elicitation and analysis phase of requirements engineering (Sommerville,

1 Many popular agile software development methodologies, which try to solve common problems of traditional methodologies, e.g. Scrum (Schwaber, 2004), have been developed for co-located stakeholders.

2007), which is the main research focus of this work. Elicitation deals with discovering the needs of all relevant stakeholders (Hickey and Davis, 2004). These needs lead to requirements, which are collected and further analyzed. The analysis activities comprise classification, prioritization, and negotiation of requirements. Different objectives of different stakeholders must be taken into account.

Requirements elicitation and analysis are activities, which often cannot be considered separately because of their highly iterative nature (Geisser, 2008). In agile software development, these activities are even more intertwined, overlapping and are repeated because agile development processes imply iterative activities (Lee and Xia, 2010; Paetsch et al., 2003). Requirements elicitation and analysis are communication-intense and critical activities in software development projects (Zowghi and Coulin, 2005). They concentrate on the task of finding out *"what problem needs to be solved"* rather than on *"how the problem should be solved"* (Nuseibeh and Easterbrook, 2000). Due to the required communication intensity of these activities and the fact that stakeholders are located at different places and possibly reside within different cultural environments, these activities are even more difficult.

1.2. Research Focus and Objective

The research objective of this work is to develop a method and a software artifact to support the activities in the early requirements engineering phase in order to overcome some of the difficulties and improve the quality of the requirements, which should eventually lead to better software products. There are many techniques, which can be used in this early phase, to elicit requirements in software development, e.g. brainstorming, interviews, business process modeling or observation (Zowghi and Coulin, 2005). A technique which has been

shown to be very effective in this phase is graphical user interface (GUI) prototyping (Gomaa and Scott, 1981; Bäumer et al., 1996; Newman and Landay, 2000; Ravid and Berry, 2000). For instance, in a study by Keil and Carmel (1995), GUI prototyping was one of the most effective customer-developer link. GUI prototypes are also popular in practice, especially in agile projects as they are quickly created and convey ideas without a lot of documentation. The prototypes show characteristics which makes GUI prototyping a promising technique for knowledge transfer between stakeholders (see Section 3.1.1). Hence, GUI prototypes have been chosen as the research focus of this thesis.

These prototypes, also known as mockups, help to visualize the system which has to be build, support the users of this system to get a clearer picture of their requirements, and serve as a tool to quickly try out various ideas and functionalities. They are an early representation of the user-visible part of the system and help to provide a common understanding of the requirements of the system, both on the developer and the user side, and improve the usability of the product (Arnowitz et al., 2007). User interface prototypes exist on a multitude of sophistication levels: from simple, paper-based sketches to dynamic functional graphical user interfaces created with heavy weighted interface creation applications (Rudd et al., 1996). Especially in the very first phase of the development, often simple, paper-based prototypes are used in order to visualize and try out ideas quickly (Chamberlain et al., 2006).

As prototypes are means to convey design ideas and help stakeholders to get a common understanding of the requirements, GUI prototyping also seems to be a promising approach in *distributed* software engineering, where communication and knowledge management problems are even more striking than in co-located settings. Prototyping can help distributed stakeholders in the early phase of development to get a common understanding of the requirements of the software product and the business context. Prototypes facilitate the communication of

stakeholders, who are situated in different locations, with different professional and cultural environments.

Furthermore, research in this area is of high relevance as prototyping is a common technique in agile development (Paetsch et al., 2003) and there are attempts to use agile development methods also in distributed environments (Ramesh et al., 2006; Hildenbrand, 2008). Nevertheless, little research has been conducted so far in the application of GUI prototyping in distributed software development. However, for the effective application of this technique, methods and tools for the collaboration on user interface prototypes over distance are needed. When designing such methods and tools, theoretical knowledge should be utilized to make them more effective. The objective of this thesis is the design and implementation of such a method and tool. The design should be informed by theory and the resulting artifacts be evaluated for their effectiveness and efficiency.

This objective leads to the following research questions:

1. How can GUI prototypes be collaboratively created, shared, maintained, and integrated into the distributed development process?

2. What kind of tool should be applied?

3. Does this approach help the stakeholders to better understand the requirements and collaborate more efficiently?

Question one focuses on a method for distributed GUI prototyping. Such a method should provide guidelines on how to create the prototypes and collaborate on them from various distributed locations, how to use prototypes of different levels of detail and functionality, and how to integrate the prototypes with other artifacts into the distributed development process. This question will be answered in Section 4.1.

The second question is related to the tool which is required to implement this method. This tool must provide the functionality that is required by the method. Beyond the design implications, which are derived from theoretical considerations and practical requirements, technical design decisions for the implementation have to be made. This question will be answered in Section 4.2.

Question three indicates that one objective of this research is to support stakeholders in the understanding of the software requirements. Furthermore, the collaboration between the stakeholders should become more efficient. To assess if the developed artifacts improve the understanding and the efficiency, they are evaluated. Chapter 5 describes this evaluation and reports its results and thus answers question three.

1.3. Research Method

To answer the questions stated in the previous section, this research will follow a design science research method (March and Smith, 1995; Hevner et al., 2004). This approach is based on a problem solving paradigm (Hevner et al., 2004). The goal is to solve the problem presented above. It is the support and improvement of user interface prototyping in globally distributed software development. To achieve this goal, two artifacts are designed, which address the problem.

The artifacts are a method and a tool, which support stakeholders in globally distributed software projects to collaborate on GUI prototypes. Following the design science approach, this research draws from existing literature and is informed by theoretical concepts, which are used for the creation of the purposeful artifacts (Hevner et al., 2004; Markus et al., 2002). These theoretical concepts

are described in Section 3.1. The theoretical knowledge is used to derive design implications for the artifacts. Further design implications from practice are determined by conducting expert interviews. These design implications are described in Section 3.2. After collecting the design implications, the artifacts are constructed. For the tool, further technical design decisions are made and the tool is implemented as a software application.

For the artifacts, which are the contributions of this research, to be purposeful, they must provide utility, which must be assessed (Hevner et al., 2004). For this purpose, the utility, quality, and efficiency of the artifacts are evaluated with well-established evaluation methods. This evaluation provides feedback for the construction of the artifact. The two artifacts of this research will be evaluated by the means of expert interviews. This is described in Chapter 5.

1.4. Organization of this Thesis

The rest of this thesis is structured as follows: Chapter 2 introduces fundamental concepts, which are important for the understanding of this research. It explains Requirements Engineering and its activities, the challenges of software engineering in distributed environments, GUI prototypes, and design science research. After this, related research is described.

Chapter 3 uses theoretical knowledge to derive design implications for the artifacts. First, the concept of boundary objects and the two theoretical concepts, the Cognitive-Affective Model of Organizational Communication for Designing Information Technology (IT) and the Media Synchronicity Theory, are introduced. After this, design implications are derived and preliminary assessed by experts from the software industry. Finally, existing tools are analyzed and it is checked if they meet the requirements of the design implications.

Chapter 4 outlines in detail the artifacts that have been developed: a method and tool for distributed GUI prototyping. The method, its steps, and example scenarios are described. Subsequently, the tool, its usage, its technologies, and its architecture are explained in detail.

In Chapter 5, the evaluation of the artifacts developed earlier is described. First, the context of the evaluation is introduced and two propositions are derived. Thereupon, the design of the evaluation is described and its results are reported.

Chapter 6 discusses the contributions, the implications and the limitations of this work. Finally, Chapter 7 offers a summary of this thesis and an outlook for future research.