Involving End Users in Distributed Requirements Engineering

Steffen Lohmann, Jürgen Ziegler, and Philipp Heim

University of Duisburg-Essen,
Lotharstrasse 65, 47057 Duisburg, Germany,
{lohmann, ziegler, heim}@interactivesystems.info

Abstract. Active involvement of end users in the development of interactive systems is both highly recommended and highly challenging. This is particularly true in settings where the requirements of a large number of geographically distributed users have to be taken into account. In this paper, we address this problem by introducing an integrated, web-based approach that enables users to easily express their ideas on how the interaction with a system could be improved. In addition, the user input is contextualized, allowing for highly structured means to access, explore, and analyze the user requirements.

Keywords: Distributed Requirements Engineering, User Involvement, Global Software Development, Web-based Participation, Distributed Participatory Design.

1 Motivation

The active involvement of end users in the analysis and design of interactive systems has become to be known as *Participatory Design (PD)* [8][12]. Over the years, a couple of methods, techniques, and tools have been developed to support PD [2]. However, though most of these approaches work well for co-located stakeholders, they lack supporting the engineering of interactive systems where needs and desires of a large number of geographically dispersed users have to be met. As this becomes increasingly common in a globalized world, PD has to face new challenges. This is addressed by the emerging research area of *Distributed Participatory Design (DPD)* [3] which investigates PD with regard to physical, temporal, and organizational distribution.

Against the background of DPD, we are working on methods and tools that support end user participation in distributed requirements engineering within the *SoftWiki* project [13]. In the following, we present an approach for the elicitation of user requirements in the evolutionary development of interactive systems. It enables distributed users to express requirements on basis of their interaction experience. We first give a brief overview on related work. Then, we describe the overall approach for distributed elicitation of user requirements, its implementation, and the underlying model. Subsequently, we provide a short insight into different ways to access, explore, and analyze the gathered user requirements and into the tools and

P. Forbrig and F. Paternò (Eds.): HCSE/TAMODIA 2008, LNCS 5247, pp. 221–228, 2008.

http://dx.doi.org/10.1007/978-3-540-85992-5_20

visualizations we are currently working on to support these activities. The paper ends with a short discussion and an outlook on future work.

2 Related Work

Research regarding the participation of distributed end users in the development of interactive systems is – next to DPD – mainly conducted under the terms of *Distributed* or *Global Software Development (DSD, GSD)*. The majority of the existing approaches are of a very general nature in that they consider support for as many stakeholder groups as possible rather than focusing specifically on the end user's expectations and needs. Furthermore – though the quantity is reduced – most approaches still heavily rely on physical meetings and direct communication (cp. e.g. *ARENA* [5] or *DisIRE* [4]).

Other attempts try to equip the users with extensive possibilities to annotate or even design the interface. Moore [10], for instance, proposes the use of GUI elements without functionality to allow end users to express their requirements: The users are enabled to create "mock user interface constructions" and augment them with textual descriptions. However, it cannot be expected that users normally have the time and skills to develop GUI proposals without any guidance. Thus, this approach is not feasible in most situations with large, distributed user groups.

A more promising approach for distributed settings is to allow users to express requirements on the basis of an existing application or test prototype. One possibility is *Digital Annotation (DA)*: Tools such as *Annotate!Pro* [1] can be used to enable end users to express requirements intuitively by annotating applications using free-form drawing and send in a snapshot of their annotations to the developing team. Rashid et al. [11] present a solution that specifically aims to support end user participation in requirements elicitation by providing a DA toolset and some predefined templates that have been developed with the needs of requirements formulation in mind.

Though DPD approaches that are based on DA can be very intuitive, they still require some effort and skills of the users in expressing their requirements and demand time-consuming interpretation in the course of analyzing and understanding the annotated screens. The possibilities for structured access or machine readability are very limited. It is nearly impossible, for instance, to automatically detect similar or identical requirements. For these reasons, the mentioned approaches do not sufficiently support settings with large user groups and provide only limited means for developers to explore, filter, and evaluate the collection of user requirements.

3 Web-based Elicitation of User Requirements

Our approach focuses on the elicitation of user requirements for web-based systems. The implementation is seamlessly integrated into the end user's web browser and provides some advantages compared to stand-alone applications:

1. A contributing user does not have to change the environment. He can express his requirement immediately when it occurs while interacting with the system.
2. Parts of the usage and system context can be captured along with the user requirement, allowing for more structured means to analyze and utilize the requirements as well as a better understanding of their intended meaning.
3. The user is enabled to explicitly point at parts of the interface the requirement refers to. Thus, requirements can be directly linked to the application structure.

3.1 Scenario and Application

The general idea of our approach is best illustrated by a brief scenario that uses the application we developed for the elicitation of user requirements (cp. Fig. 1).

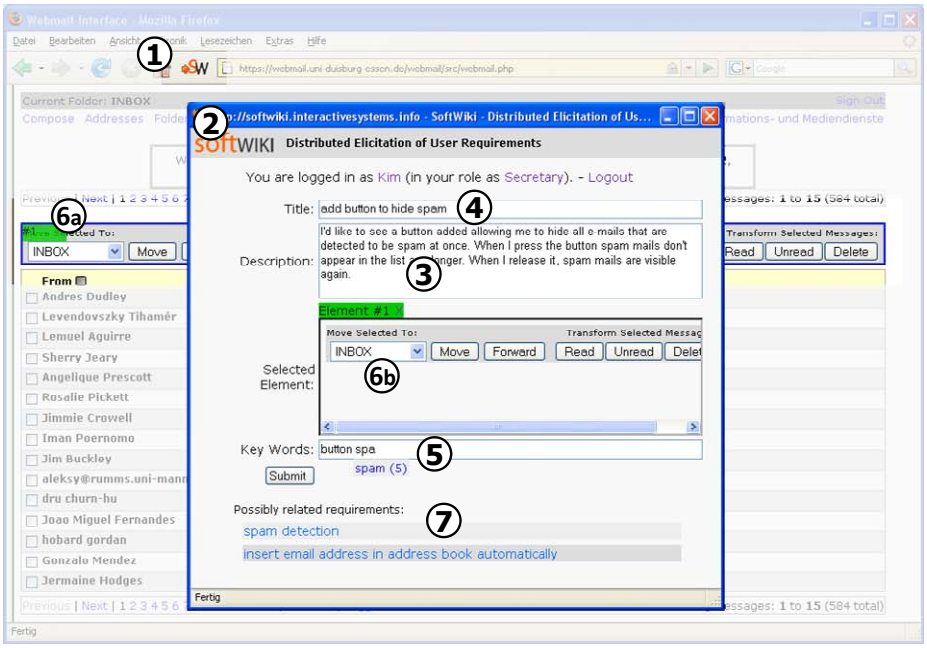

Fig. 1. Web-based interface for the submission of requirements

Imagine an employee who uses the company's web-based mail application in her daily business. While checking her e-mails, the employee misses a feature that she would like to see realized in one of the next releases, in this case, a possibility to hide all e-mails that have been detected to be spam at one click. Thus, she presses a button that is integrated in the user interface of her web browser (1). A pop-up window appears (2) containing a web form where she enters a description of her requirement (3) and optionally adds an adequate title (4) and some keywords (5). If her requirement refers to an element of the visible web page, she does not have to textually describe the element but can directly point to it as follows: While the pop-up

window is opened, selected elements are highlighted (6a) and can be copied to the web form (6b) simply with a click[1]. Finally, the employee submits her requirement, receives a confirmation, the pop-up window closes and she returns to the web application where she continues to check her e-mails.

The user interface is reduced to its essential elements so that it is immediately understandable, minimizes user effort and hence encourages participation. In the example given, the user does not even have to classify her requirement into a pre-defined taxonomy or a collection of existing requirements but is simply asked to provide some meaningful, freely chosen keywords. In order to further ease participation, requirements that are identified as similar to the one the user is entering are displayed below the web form (7)[2]: That way, the user does not need to formulate a requirement a second time that already exists. Furthermore, the amount of redundant requirements is reduced, leading to lower effort in analyzing the requirements.

3.2 Conceptual Model and Gathered Information

The conceptual model underlying our approach is shown in Fig. 2. It is divided into four parts. As is common in requirements engineering, the *basic data* consists of a *description* and *title* of the requirement that in our case are formulated by the user, and an automatically assigned *identification (ID)*. An automatically generated full-*text index* of the title and description together with the user added *keywords* ease access and are used for searching in the requirements and calculating the similarity measure, amongst others.

Along with the user input, additional information regarding the *usage* and *system context* is captured. Following Kaltz et al. [6], we break down the *usage context* into the facets *User & Role*, *Location*, *Time*, *Device*, and *Task*. Technically, this context data is derived via several mechanisms using header information of the transfer protocols and additional information that is gathered and sent by the web browser plug-in in combination with user profiles, geolocation, and lookup tables[3]. The context facet *User & Role* takes into account that a user may want to be able to state requirements out of different roles in some situations – for instance, a director of a company might also want to express a requirement from his perspective as an ordinary user of the system.

The *Task* facet of the usage context is highly related to the concepts *System State* and *System Pointer* that together form the system context. The former expresses that each user requirement occurs within a specific state of the system that it can be linked with[4]. The latter represents elements that the user explicitly refers to when formulating his requirement (see Sec. 3.1). Depending on the implementation of the

[1] Hyperlinks in the web page are temporarily deactivated for this purpose.

[2] The similarity measure is calculated in the background while the user types in her requirement using asynchronous server requests as well as statistical and linguistic algorithms.

[3] Depending on the particular use case, the derived context data cannot be expected to be perfectly correct.

[4] The state of the system and usage context when the requirement is entered by the user is, strictly speaking, not necessarily identical to that when the requirement occurs; but in most situations this is likely to be the case.

web application and its internal structure, these general model concepts can be further broken down and filled accordingly. For instance, if a model-driven web engineering approach [9] has been used to develop the application, links between requirements and parts of the system models can be explicitly set. However, this requires that the models are accessible by the web browser plug-in at runtime and that the corresponding system state is represented in these models accordingly.

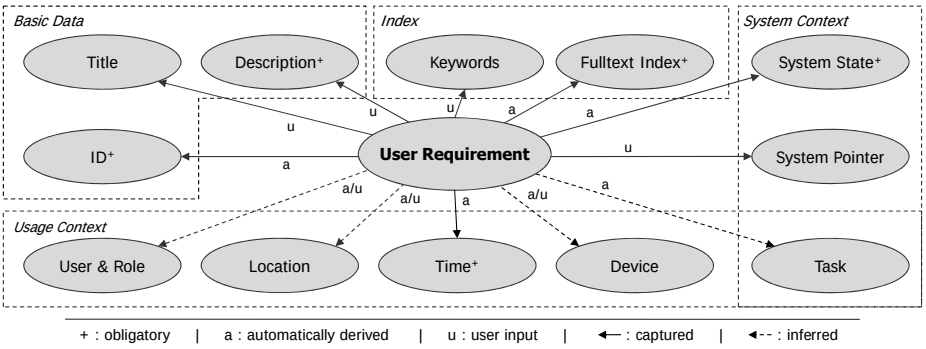

Fig. 2. The user requirement is linked to a number of model concepts

Our basic implementation follows a generic approach to determine the system context: The *System State* is derived from the URL of the corresponding web page and variables provided by the transfer protocol. The *System Pointer* consists of the paths of the selected elements according to the *Document Object Model (DOM)*. Of course, the expressiveness of relations to the system context is limited in this generic approach.

As already mentioned, information regarding the task that the user performs when the requirement occurs might be inferred from the system models to some degree. Such as in case of the other model concepts, the *Task* concept can be further broken down depending on the implementation of the specific use case.

4 Analyzing the User Requirements

The information that is captured along with each requirement provides multiple ways to access the requirements collection and thus eases its exploration and analysis. Again, this is best illustrated by an example:

Figure 3a shows a screenshot of a web-based prototype we developed for analyzing the requirements that are elicited on the basis of the tool and underlying model described in Sec. 3. The interface consists of a main view on the requirements collection and a sidebar containing visualizations that offer various options for filtering the requirements according to the model facets. Two visual filters are implemented in the current prototype: a map visualization that shows the requirements according to their geo-coordinates and can be used for location-based

filtering, and a tag cloud visualization that alphabetically lists the user-assigned keywords, each with a size that corresponds to its usage frequency.

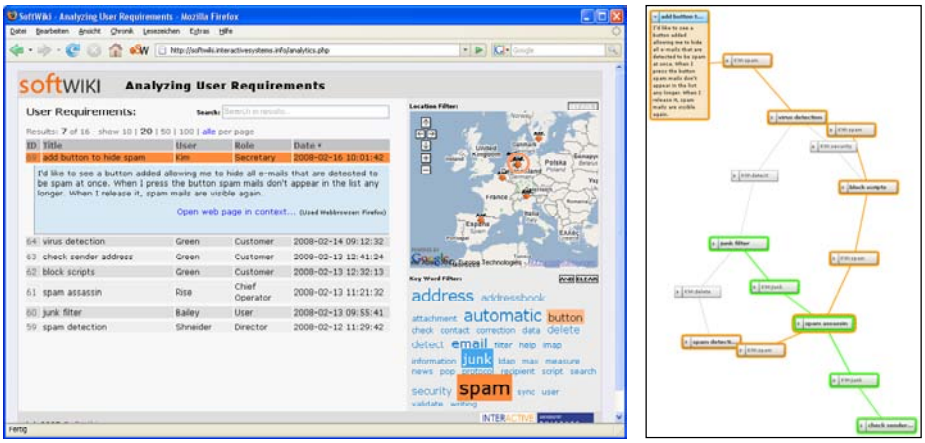

Fig. 3. a) Web interface for the analysis of user requirements, b) Alternative graph visualization

Continuing the scenario outlined in Sec. 3.1, one of the developers of the company's webmail application is going to define specifications for the next software release. Assume that the developer is particularly interested in all user requirements regarding the topic spam that have been stated by the employees working at company sites in Europe. He thus changes the visible area of the map so that only Europe is shown. Then, he selects all keywords that deal with the topic spam – in this case he chooses the synonyms spam and junk – and defines that these keywords should be interconnected by the logical OR operator. As a result, only requirements are listed that meet these criteria. Furthermore, the developer can take a look at the contextual situation as it was when a specific requirement has been entered by an employee. For this purpose, a pop-up window simulates the presentation of the application according to the context data that has been captured along with the requirement (e.g., the corresponding web page, the size of the content area, etc.). If references to parts of the web page were set by the employee (see Sec. 3.1), these elements are highlighted in the simulated view[5].

Besides the map and tag cloud visualization, further filter options are possible according to the model presented in Sec. 3.2. Furthermore, we investigate alternative ways of presenting the requirements collection in the main view. Figure 3b shows a prototype of a graph visualization that helps the analyst to reveal relationships, similarities, and conflicts between requirements that are otherwise hidden in the list view.

[5] In our current prototype, the highlighting works properly only in cases where the DOM path to the selected web page elements remains the same.

5 Discussion and Future Work

The presented approach differs from related work in that it is integrated into the user's web browser. That way, the user does not have to change the environment to express an idea on how the interaction with a web-based system could be improved. The user input can furthermore be related to the system and usage context. This is possible mainly due to the fact that web applications are predominantly based on script languages that are interpreted by the web browser at runtime. However, with the advent of user interface markup languages for operating systems (e.g., XAML [14]), the application of a slightly adapted approach is also increasingly feasible outside the web browser.

We paid special attention to the balancing of our approach by minimizing the effort for users who express requirements and, at the same time, capturing sufficient metadata to enable structured analysis and further processing of the requirements. This is best demonstrated by the possibility to select interface elements a requirement refers to (see Sec. 3.1): On the one hand, the user does not have to describe GUI elements but can simply point at them, and, on the other hand, the developer does not have to guess what element is meant and can work with the reference, for instance, aggregate all requirements that have been related to one GUI element.

Generally, the presented approach can be used in all settings where users shall be enabled to give feedback regarding a web-based system, ranging from feature requests and bug tracking to remote usability testing. The overall aim is to establish a closer relationship between users and developers in settings with large and distributed user groups. First tests showed that the general approach and the developed tools are quickly understood by users. Currently, we are preparing a comprehensive case study that examines the developed applications within the larger context of the *SoftWiki* project.

In order to increase participation and create an awareness of what happens with the user input, we are investigating different requirements tracking, user feedback, and gratification mechanisms. In addition, we study how users might discuss, reformulate, or vote for a requirement that has already been stated by someone else and is identified as possibly related.

References

1. Annotate!Pro, http://www.annotatepro.com/ (2008/Jun/11)
2. Bødker, K., Kensing, F., Simonsen, J.: Participatory IT Design – Designing for Business and Workplace Realities. MIT Press, Cambridge, MA (2004)
3. Danielsson, K., Naghsh, A.M., Gumm, D., Warr, A.: Distributed Participatory Design. In: Extended Abstracts of the 2008 Conference on Human Factors in Computing Systems (CHI '08), Florence, Italy. ACM, New York, NY (2008) 3953–3956
4. Geisser, M., Heinzl, A., Hildenbrand, T., Rothlauf, F.: Verteiltes, internetbasiertes Requirements-Engineering. Wirtschaftsinformatik 49(3) (2007) 199–207
5. Grünbacher, P., Braunsberger, P.: Tool Support for Distributed Requirements Negotiation. In: Cimititle, A., De Lucia, A., Gall, H. (eds.) Cooperative Methods and Tools for Distributed Software Processes. FrancoAngeli, Milano (2003) 56–66

6. Kaltz, J.W., Ziegler, J., Lohmann, S.: Context-aware Web Engineering: Modeling and Applications. Revue d'Intelligence Artificielle 19(3) (2005) 439–458
7. Kaser, O., Lemire, D.: Tag-Cloud Drawing: Algorithms for Cloud Visualization. In: Proceedings of the WWW 2007 Workshop on Tagging and Metadata for Social Information Organization, 2007
8. Kensing, F., Blomberg, J.: Participatory Design: Issues and Concerns. Computer Supported Cooperative Work 7 (1998) 167–185
9. Moreno, N., Romero, J.R., Vallecillo, A.: An Overview of Model-Driven Web Engineering and the MDA. In: Rossi, G., Pastor, O., Schwabe, D., Olsina, L. (eds.) Web Engineering: Modelling and Implementing Web Applications. Springer, Heidelberg (2008) 353–382
10. Moore, J.M.: Communicating Requirements Using End-User GUI Constructions with Argumentation. In: Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE'03), Montreal, Canada. IEEE, Washington (2003) 360–363
11. Rashid, A., Meder, D., Wiesenberger, J., Behm, A.: Visual Requirement Specification in End-User Participation. In: Proceedings of the 1st International Workshop on Multimedia Requirements Engineering. IEEE, Washington (2006)
12. Schuler, D., Namioka, A.: Participatory Design: Principles and Practices. Erlbaum, Hillsdale, NJ (1993)
13. SoftWiki – Research project, funded by the German Federal Ministry of Education and Research (BMBF). For more information, see http://softwiki.de/
14. Extensible Application Markup Language (XAML), see e.g. http://msdn.microsoft.com/en-us/library/ms752059.aspx (2008/Jun/11)