Fostering Remote User Participation and Integration of User Feedback into Software Development

Steffen Lohmann
University of Duisburg-Essen
Interactive Systems and Interaction Design
Lotharstr. 65, 47057 Duisburg, Germany
steffen.lohmann@uni-due.de

Asarnusch Rashid
Research Center for Information Technology
Information Process Engineering (IPE)
Haid-und-Neu Str. 10-14, 76131 Karlsruhe, Germany
rashid@fzi.de

ABSTRACT
Permanent involvement of end users in software development is both highly recommended and highly challenging. Against the background of our results and experiences from two research projects, we summarize several key issues and design concerns that need to be considered when integrating users and their feedback into software development.

Categories and Subject Descriptors
K.6.3 [Software Management]: *Software development.* D.2.1 [Requirements/Specifications]: *Elicitation methods*. D.2.2 [Design Tools and Techniques]: *User interfaces*. H.5.2 [User Interfaces]: *User-centered design.* H.1.2 [User/Machine Systems]: *Human factors.* I.3.6 [Methodology and Techniques]: *Interaction techniques.*

General Terms
Design, Human Factors

Keywords
Remote User Participation, User-centered Software Development, Distributed Participatory Design, User Interface Annotation

1. INTRODUCTION
Nowadays, software development is increasingly characterized by evolutionary processes and short development cycles. Modern software systems usually need continuous updating, improvement, and customization. Perpetual usability evaluations and user surveys are crucial to guarantee that a software system meets the users' needs. Development concepts such as *Participatory Design in Use* [2] emphasize the importance of continuous user participation. However, the spatial and temporal distribution of system users often limits the possibilities for co-located methods of participatory design. In many cases, user participation is only remotely possible, e.g. via computer-mediated forms of communication.

I-USED'08, September 24, 2008, Pisa, Italy

Within the research projects *SoftWiki* [8] and *CallaBaWue* [3], methods and tools have been developed that ease remote participation of end users in the software development process, in particular with respect to requirements elicitation and usability evaluation. The basic toolset in both projects consists of a collaboration platform and participation channels that enable users to make suggestions for improvements concerning a certain software product (cp. [4, 6]). During the development of these methods and tools as well as in three usability tests and two case studies (one short-term and one long-term) including over 50 participants in total, we got valuable insights regarding successful forms of remote user participation as well as drivers for the integration of user feedback into software development. In the following, we summarize some key issues and design concerns that need to be taken into account when involving distributed users in software development.

2. DIMENSIONS FOR REMOTE PARTICIPATION
Several important issues and conceptual aspects regarding the integration of distributed users to improve software systems, such as the reporting of bugs or remote usability evaluations, are discussed in related work (e.g., [5, 1, 2]). On a general level, we identified three dimensions that appeared to be central when it comes to the implementation of computer-mediated user participation in software development: *degree of autonomy*, *number of users*, and *level of collaboration*.

The degree of autonomy can be divided in the two opposite approaches of *autonomous* and *event-driven participation*. Autonomous participation means that the user decides on his own when to participate. A typical scenario would be that the user expresses requirements whenever they appear in his daily use of a software system. Event-driven participation forms, in contrast, explicitly invite users to participate in certain situations or at particular points in time. Our favorite solution is a combined approach that regularly reminds the users that they can influence the system design and inspires participation by providing certain topic frames and at the same time allowing them to contribute at any time, independently of the particular development status. Especially the last aspect seems to be crucial, since we experienced that test users very much liked the possibility of being able to express requirements immediately whenever they occur while interacting with the system.

In addition, the optimal form of participation support relies largely on the *number of users* that are expected to be actively

involved in the development. The higher the number of participants, the more important are mechanism that guarantee systematic and structured elicitation and analysis that can handle large amounts of requirements. For instance, within the tool *OpenProposal* [6], we made promising experiences with the digital annotation of user interfaces that are automatically saved as screenshots and send to the developers. In cases where large numbers of users participate, automatic utilization of user annotations proofed to be useful as in the tool *Softfox* [4] that supports direct linking of user input to the application structure or underlying system models, in particular if a model-driven development [7] approach is being used.

A third design dimension concerns the *level of collaboration*, both among users and between users and developers. A central question is whether users provide requirements individually and independently or if the requirements are collaboratively developed and improved. Within our research, we came to the result that sophisticated solutions should collect the user at the point he is willing to participate. For instance, a web platform can provide collaboration support such as commenting, discussion, or cooperative editing features as well as possibilities to rate, vote, and link requirements. Embedded participation channels can be moreover provided for those users who are willing to participate but are not willing to deal with the collaboration platform. However, some kind of awareness regarding already existing requirements should be given in any case – this reduces the effort for the user as he does not need to formulate a requirement a second time that has already been expressed. Furthermore, the amount of redundant requirements is reduced leading to lower effort in analyzing the requirements.

3. FURTHER ASPECTS

Next to these basic design dimensions, we identified several aspects that we regard as highly valuable for successful implementation of remote user participation in software development. In the following, we summarize further key issues that can help to lower the participation barrier and to better link user feedback with the software product.

3.1 Reducing the Participation Barrier

Integration into the user's environment: The participation interfaces should at best be directly embedded into the user's system environment to establish an affordance always reminding the user that involvement and thus system improvement is possible. For instance, some kind of 'participate'-button can be constantly visible on the desktop or can be integrated into the interface of the web browser or application of interest.

Lightweight Participation: It should be possible for the user to participate whenever an idea for improvement comes to his mind, resulting in only a marginal interruption of his actual activity or workflow. At best, the user should decide what and how much information he wants to provide. The initial input should be based on a lightweight and informal process that can later be refined and elaborated.

Simplicity and Assistance: All interactions with the user interface should be as simple and self-explaining as possible in order to encourage users getting involved. The interface should not require to login each time the users express a requirement; appropriate interaction support, such as automatic form filling or system suggestions, should moreover be provided. The user should furthermore not be enforced to provide extensive data or make classification decisions that are cognitively challenging. However, too much assistance, such as pre-defined templates or automatic system proposals, can also have a negative impact on the creativity of the user.

Transparency: In every situation, it must be clear to the user what data is captured along with his input. Ideally, the user can continuously track the progression of his requirements in the development process. The user's motivation is heavily based on the fact that he recognizes how the system is improved as a consequence of his input, which might lead to a personal benefit.

3.2 Linking User Input to Software Artifacts

Most user requirements refer to specific artifacts of the software system. We found that both – users and developers – can benefit from options allowing to implicitly or explicitly link requirements to parts of the software system.

A key feature of our tools that has been rated as highly valuable in user tests is the possibility to directly refer to elements of the graphical user interface while formulating requirements. This is either realized by digital annotation (in case of *OpenProposal*) or by direct selection of web elements (in case of *Softfox*). The assumption of this feature is that many software artifacts have a representation in the user interface, in particular artifacts that end-users refer to. On the one hand, references to the user interface ease the requirements formulation for the user as he does not need to textually describe the interface elements but can directly point at them. Furthermore, this concretizes and illustrates his ideas for improvement and can reduce typical problems that often arise from text-only communication such as misconceptions due to wrong word choice, incomplete data, or descriptions that are too elaborate. On the other hand, the application context can provide valuable assistance in systematically analyzing the user requirements; the analyst can, for instance, inspect all requirements at once that refer to a certain element of the user interface.

4. CONCLUSION AND OUTLOOK

This position paper reported several aspects we experienced as valuable to foster user participation in distributed settings and help to integrate feedback in the software development process. However, we have not discussed in what ways developers have to rethink and change their habits to make remote user participation successful. This remains a topic for future work.

5. REFERENCES

[1] Castillo, J.S., Hartson, H.R., Hix, D. 1998. Remote Usability Evaluation: Can Users Report Their Own Critical Incidents? In CHI'98 Human Factors in Computing Systems, 253-254.

[2] Draxler, S., Stevens, G. 2006: Getting Out of a Tailorability Dilemma. In Informatik 2006 – Informatik für Menschen, 1, LNI P-93, 576-579.

[3] CollaBaWue – research project, funded by the Landesstiftung Baden-Wuerttemberg Foundation, see http://www.collabawue.de/

[4] Lohmann, S., Ziegler, J., and Heim, P. 2008. In Engineering Interactive Systems 2008, LNCS 5247, 221–228, in press.

[5] Nichols, D.M., McKay D., Twidale, M.B. 2003. Participatory Usability: Supporting Proactive Users. In Proc of 4th ACM SIGCHI NZ Symposium on Computer-Human Interaction (CHINZ'03), 63-68.

[6] Rashid, A., Wiesenberger, J., Meder, D., Baumann, J. 2008. In Proc of the PRIMIUM Subconference at the Multikonferenz Wirtschaftsinformatik (MKWI), CEUR-WS 328.

[7] Schmidt, D.C. 2006. Model-Driven Engineering. IEEE Computer 39(2).

[8] SoftWiki – research project, funded by the German Federal Ministry of Education and Research (BMBF), see http://softwiki.de