Framing Participatory Design Through e-Prototyping

Wolf-Gideon Bleek, Martti Jeenicke, Ralf Klischewski
Software Engineering Group, Department for Informatics, University of Hamburg
Vogt-Kölln-Str. 30, D-22527 Hamburg, Germany, +49 40 42883-2307
{bleek, jeenicke, klischewski}@informatik.uni-hamburg.de

ABSTRACT
The paper discusses how a new way of prototyping can serve as a method to support a participatory and evolutionary design approach within Web projects. “e-prototyping” is meant to frame participation of Web users and other stakeholders in the design process through providing and maintaining a variety of communication channels for (user) feedback on frequently released software versions as well as establishing a steering board which takes into account the users voice in sorting out the feedback and setting priorities for the following design effort. From the software process perspective, e-prototyping supplies the development arena with the information needed (i.e. requirements), thus embedding the design activities in a loop of continuous communication and learning.

Keywords
Participatory design, e-prototyping, Web projects, software development

INTRODUCTION
Prototyping has become a well established method within participatory software design (we use the term design to embrace also software development and deployment), even though it does not resolve all the difficulties on the way of successfully integrating the users’ perspective in software processes. However, as software design projects are increasingly focussing on Web-based applications (such as in e-business, e-government etc.), a number of new circumstances increase the difficulties of applying prototyping as a method within participatory design (PD):

- Use situation: As new technology enables new forms of IT use, the relation of Web-based software and its users is much less tangible. Users are often unknown, and other products/services are “only a mouse click away”. Web users often have more choices on which application they want to use, but less choices how they want to use it (e.g. users have no control concerning software updates, configuration etc).
- Software process: Content and style are, for Web users, at least as important as the tool functionality. Therefore, the development of Web-based applications involves different kinds of experts and professions (beside the software engineer). To keep Web users attracted, most design projects strive for short term innovation whereas project ending is often not predefined.
- Organizational environment: Web projects are usually embedded in far reaching networks of stakeholders crossing many organizational boundaries. At the same time, cooperative relations are less obliged and partners frequently drop out, become substituted, or the network extends more and more.

Taking this into account, we find that “traditional” prototyping is based on assumptions which are not (or only partially) valid for the development of Web-based applications (regarding e.g. actors involved, organizational frame, communication, time frame, controllability and relevance of application). From the developer’s point of view, as the frame for the user-designer-interaction through prototyping seems to dissolve, one needs to ask:

- How can prototyping still support participatory design in the world of networked and distributed systems?
- What kind of modifications are necessary in the management of software design projects?
- How can the organizational environment of design processes still accommodate participation in bringing out Web-based applications?

In this paper we seek to frame PD through (1) analyzing the gathering of requirements and the involvement of different actors in developing Web-based applications, and (2) proposing to integrate evolutionary software development and prototyping based collecting and evaluating feedback from users and other relevant actors in “productive mode”, accompanied by various communicative and organizational measures to ensure that the users voice has a say in the design process.

IN SEARCH FOR FRAMING PD IN WEB PROJECTS
Prototyping has been discussed extensively in application oriented software engineering as well as in PD. From the software developer’s perspective, e.g. Sommerville ([13], pp. 138-153) describes it as a means of requirements analysis and validation. Prototypes support the communication among developers and users, by enabling them to “experiment with requirements”.

In addition to the developer centred methods, cooperative prototyping [4] is an approach in which the process of creating and evaluating prototypes is seen as a cooperative ac-

tivity between the future users and the designers of a product. By letting users play around with prototypes in simulated or real use situations, problems that occur can be analysed and solved. This way "users can participate actively in improving the prototype" ([3], p.170). Cooperative prototyping puts an emphasis on the learning aspect of the process – prototypes are therefore an important type of artefact and a source of insight in a continuous learning process.

From an participatory design point of view, future users as well as other actors should be well integrated also in the development of Web-based applications. However, there are several problems with implementing PD principles in this domain. In this paper we focus on the development of Web-based applications and services for unknown user groups. We will not discuss corporate intranets and extranets in which developers can relate to a well defined group of users and other actors (for intranet development see e.g., [5],[11],[12]). Most of our findings will also apply to e-commerce systems (comprising classical shop and auction systems), although we will not address the specifics of workflow and transaction management.

In this section we point at the new conditions, difficulties, and challenges of gathering requests/requirements and communicating with the relevant actors. Our findings are mainly based on our involvement and experience in two projects: the development of CommSy, a web-based application designed to fit the needs of project-based learning groups [10], and of www.hamburg.de, a city portal and interactive Web-site [2]. Our evaluation of the project cases showed that the following questions arose repeatedly and turned out to be crucial from the point of view of software development:

- How can the (initial) requirements for network-based Web applications be defined?
- How can the requirements be gathered systematically, if the target users of the system (the Web users) are unknown and hardly describable in their characteristics?
- Which actors should participate and how?
- What are the consequences for requirements gathering and the development process of the continuous expansion of the technical system that supports the applications?

We claim that these problems arise in many Web projects. In "traditional" software projects requirements usually relate to milestones structuring the overall software process. Within these processes, prototypes are usually built in order to gain new insights and support decision making if applicable, embedded in the iterations of requirement analysis, prototyping, realization, releasing the product and revising. Web projects do not enjoy this kind of freedom:

- Any software released to use on the Web is without protection: publicly accessible Web prototypes are always exposed to public criticism – no more "playing around" with a development system.
- Initial requirements are defined by the providers' view for a potential application (the wishes and demands of the current user group will become evident only through the first running version of the system).
- Each feedback round with users needs time to prepare, present, communicate, and evaluate. But strong market pressure and high expectations usually do not allow Web projects to wait for this.
- Web users expect new versions regularly, especially when waiting for requested functionality.
- This leads to considerably shorter development cycles and consequently to pressure on the developers to define work packages for shorter time periods.
- What e-applications have in common is that they are "early adopters" [11] in their domain, i.e. they offer a new service on the Internet. Development has to keep in mind that the application is expected to feature high quality and innovation.

With more groups of actors being involved, recognizing and acknowledging the different perspectives becomes a crucial task for requirement gathering within projects. The relevant actors cannot be represented within a simple actor model (e.g. including developers, users, and management). Actors contributing to the system development take on new roles such as "technology champion" [5], (sub-)service provider, and others. Also, we can still identify well-known roles such as contractor (the financier of the project), user, developer and customer, but there are significant shifts in interest:

- Contractors, at least in principle, expect a return on their investment. But Web projects are often not accountable in terms of rationalization effect, the result may be an image improvement, an increase in market potential or an expansion of the service portfolio, and in many cases project investments are 'strategic' with no clear-cut criteria for evaluation available.
- With Web applications, the software users are often also the customers (or clients) of the organization contracting the development (in contrast to "traditional" projects, where the users work for an organization creating some value for external customers), thus changing the relation between the user and the contractor.
- The user group is not well defined and profiles are hard to obtain (unlike in companies, where users can be characterized by their jobs or functions).
- New roles can and should be identified, e.g. strong complainers criticizing errors or missing functions on a regular basis, volunteers trying to play an active role in the further development of the Web application by spending a lot of time on evaluation and making constructive suggestions for improvement.
- The developers' main task is to develop and integrate a system in a given environment (coping with existing or predefined technology) and to make it run reliably. The developers' perspective is focussed on keeping the software error-free, making the latest back end technology run, and implementing state-of-the-art features. Their use perspective is mostly limited to one of a "power-user".

In Web projects, the different actors with their perspectives and interests are often not part of the same organizational unit which precludes direct and personal discussions (e.g. users or decision makers cannot easily be invited or are not available for single or group interviews) or even simple user observation. In addition, without a common social frame (the organization) users are more difficult to motivate for participation: Internet users, in case they dislike the application, will just stay away, and organizational users in remote locations mostly feel they do not have a say anyway.

Furthermore, Web projects are facing new operating conditions which become visible step by step as the application is already in productive mode with reactions from "real" users. Thus requirement gathering in Web projects cannot make use of "traditional" prototyping as many presumptions no longer hold true. Bringing in the perspective of the manifold relevant actors and giving them a voice in the design process of Web-based applications needs new approaches.

HOW TO DO E-PROTOTYPING

In this section we suggest e-prototyping as an approach to create an environment for integrating users in Web projects through the use and evaluation of consecutive software versions. Firstly, from the developers' point of view, we argue that the current trend in software engineering towards shorter development cycles leads to an intertwining of prototyping and release management. Secondly, we describe the steps of our e-prototyping approach in comparison to "traditional" prototyping activities. Thirdly, we show how obstacles in the user-developer relation can be overcome by promoting and integrating communication into the development process.

Speeding up cycles in software development

Shortening development life cycles is an issue in various fields of software engineering (e.g. for the German software industry see [14]). Lately, the approach called Extreme Programming [1] has gained much attention, calling for shorter cycles on all levels of software engineering in order to increase the quality of a software product. The system should grow constantly through continuous integration and frequent releases. This approach has been applied extensively in the CommSy project, in which discussion forums were used to gather feedback, which in turn was regularly discusses in an architecture group.

Frequent releasing is also very common in projects of the open source community [9]. There, a two-way communication (user feedback and reports on development) is critical (e.g. Mozilla project), posing new management tasks to the project. This is in line with our experiences: in the CommSy project, the number of system users and of use settings increased rapidly so that the developers lost direct contact to the end-users. The hamburg.de project at first failed to produce frequent releases and user communication, which resulted (among other) in users "misusing" the guest book to file complains.

In short, evolutionary approaches and systematically making use of user feedback seem to become state of the art in application oriented software engineering. Moreover, in Web projects these development approaches have to be intertwined with the 'productive mode' of any software developed. We see e-prototyping supporting an evolutionary approach for Web projects based on short development and release cycles with each of the releases being treated as an e-prototype for the next development effort.

Steps in e-prototyping

Within evolutionary and participatory software development, cyclic approaches were suggested as early as in the 1980s, putting emphasis on the communication between developers and users. E.g. the STEPS model [7] proposes development cycles consisting of (1) revision establishment, (2) production, (3) releasing a system version and (4) application of the version. Based on this kind of approach, we propose to realize prototyping within an evolutionary Web application development process. Framed by the four steps of evolutionary prototyping [6] – functional selection, construction, evaluation, and decision on further use – we outline how to do e-prototyping (see figure 1):

1. In order to perform a **functional selection**, requirements need to be gathered. In the area of Web applications, as the user group cannot be determined beforehand (or at least only very vaguely), initial requirements have to be anticipated [8] at the beginning of the project by the members of the development teams, the (Web) provider organization, and/or the business partners. It has proved useful to gather the actors involved in a so-called "steering board" which can also include user representatives. At the beginning of development the goal in mind is "to go public" fast, to reduce the "time to market", to face the discussion with the users of the new system version within a short time, and to integrate users into the development process as soon as possible. The plan for the first usable version should cover only essential functions that can easily be handled by the developing team. Therefore an appropriate functional selection is the basis for a cyclic development. In the hamburg.de project this approach was adopted after the failure of a big-bang approach. The experiences gained during each cycle help the developing team to master further steps in the development. The functional selection in the next cycles is then based on decisions from the steering board evaluating user feedback (see below, steps 3 and 4).
2. In each cycle, **construction** focuses on technical and functional requirements selected. This way the CommSy project accomplishes to fix mostly all reported bugs so far and add some new features. After construction the software is released, i.e. made accessible for use through the Internet. It will then be treated as a productive system by the people who use it, although it is regarded as a prototype from the development perspective and used as "a learning vehicle". In contrast to "traditional" prototypes, it is being used in real life conditions, and is not

labeled as a prototype. In that respect, e-prototypes as releases must therefore meet higher standards than "traditional" software prototypes, which puts additional pressure on the developers to strive for high quality. Construction must aim at a working system as a precondition to obtaining and evaluating user feedback.

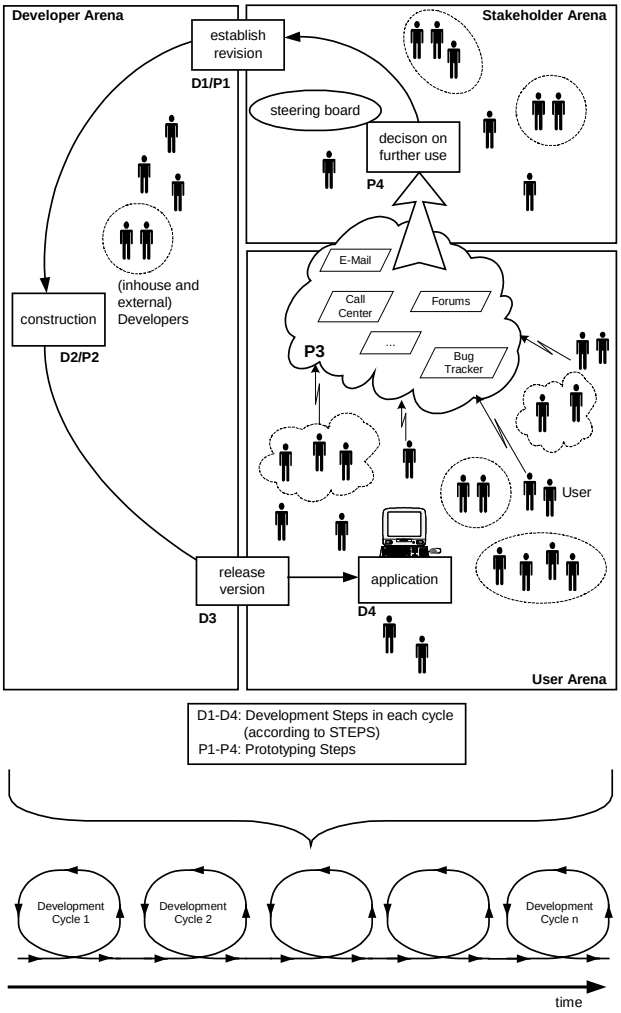

Figure 1: The e-prototyping cycle supporting evolutionary participatory design

3. The **evaluation** heavily relies on communicational means established in parallel to the use of each e-prototype/release (see also next subsection). Feedback concerning the current software version may consist of error reports collected from users and system administrators, usability problems excerpted from discussions, and additional (and new) requirements of the users (technology pull). Error reports, usability problems, and additional requirements are collected and published through diverse communication channels. E.g., channels in the hamburg.de project were the guest book and a call center, and the CommSy project utilized mainly discussion forums. Calls for new 'strategic' applications from other stakeholders to gain a competitive advantage (technology push) are also collected and discussed in the development team and the steering board.
4. **Decisions on the further use** of the software version (the e-prototype) are based on the evaluation. The decision on the further use is made from the management perspective (steering board) and is closely related to the next cycle's functional selection. It is influenced by users, providers and other stakeholders from the application domain to integrate their view into the development process. E.g. decision making took place in the steering group of the hamburg.de project and in the architecture group of the CommSy project.

These four steps can be regarded as one cycle in an evolutionary software development process framed by the prototyping approach. The decisions taken after evaluation give input for the next cycle starting with functional and technical selection prior to the construction of the next version. The requirements for the follow-up version (based on necessary corrections and selected innovative changes) should be limited in such a way that the construction and release of the next version (e-prototype, release) will not take longer than three months. The hamburg.de project first struggled with the cycles (in fact it failed to realize a big bang release) but then accommodated to releasing versions every few months. Within the CommSy project some cycles were shorter than four weeks, which made it difficult to impart those new version to the users.

Communication and Management

As communication between users and developers is essential for driving the prototyping process, we need communicational means to help establish some interaction with the (mostly) "unknown" Web users. The following channels have proved to be particularly useful: email sent to an address reserved for that purpose (e.g. feedback@web-organization.com), a call center where users' problems and suggestions can be recorded, and a Web site containing error report forms, and electronic discussion forums. As far as possible, contributions and calls have to be answered if necessary. In both our projects establishing and maintaining the channels has been a challenge of its own. Above all, the feedback from the various channels needs to be gathered consistently to support an efficient evaluation.

User participation within Web projects relates to a unique "cultural" background of the Internet community. There, users often voluntarily take an active role in a project without directly deriving any benefits from it (cf. newsnet forums), e.g. because they are interested in a particular software. For the successful interaction between developers and users, it is important that these users feel that they are taken seriously and the software provided is 'reliable' (which implies, among other things, an assurance that support is available for those voluntary users in case of a software causes serious damage on the user side).

Updates of a running e-application should be made at short intervals (3 months at maximum). Bug fixes (patches) are required more frequently to keep the above mentioned feedback channels clear of error reports. The more "buggy"

a system is, the more of the communication is about errors or the existence of bugs. In the end, only a bug-free system enables the freedom for communication about advanced functionality and usability.

Software processes management applying e-prototyping must strive for short releases, communication, and innovation. The process described is much less controllable as it is in "traditional" software development. For example, a successful application attracts more users, which leads to a greater load on the system and in turn provokes problems and erroneous behavior. As a consequence, a redesign of the system's architecture might become inevitable. Thus the emphasis of development activities can shift from a solely functional oriented approach to a structural redesign in order to meet demands of scalability and a high load service. Additional security needs on the part of the user can lead to safety features within the system initially not foreseen and planned. Also, market pressure is another factor that contributes to very short development intervals and frequent releases of innovative system versions.

To manage the outlined process, all feedback collected from the different channels must be associated with a particular version and evaluated by a steering board. They decide what to put on the development agenda. This is the foundation for the next release addressing bugs which should be removed immediately and feature enhancements. Persons reporting a bug should be told about improvements directly. It should also be made clear at what point the improvements will be integrated into the live system. In order to avoid duplicate reports, information about known problems should be available to other users.

CONCLUSION

In this paper we discussed how a new way of (e-)prototyping can serve as a method to support a participatory and evolutionary design approach within Web projects. Based on short cycles of software development and release, e-prototyping is meant to frame participation of Web users and other stakeholders in the design process through providing and maintaining a variety of communication channels for (user) feedback on frequently released software versions as well as establishing a steering board which takes into account the users voice in sorting out the feedback and setting priorities for the following design effort. From the software process perspective, e-prototyping supplies the development arena with the information needed (i.e. requirements), thus embedding the design activities in a loop of continuous communication and learning. However, future research needs to verify the hypothesis that e-prototyping provides an appropriate general frame for PD of Web-based applications and/or to analyze the methodological, organizational and political success factors for such kind of endeavors.

REFERENCES

1. Beck, K. *Extreme programming explained: embrace change*. Addison-Wesley, Reading, Mass, 2000.
2. Bleek, W.-G. Situations in Life to Support the Use and Modeling of Municipal Information Systems. In: Remenyi D. and Bannister, F. (eds.). *Proceedings of the European Conference on Electronic Government*, Trinity College Dublin, Ireland, 2001, 49-60.
3. Bødker, S., Grønbæk, K., Kyng, M. Cooperative Design: Techniques and Experiences from the Scandinavian Scene. In: Schuler, D. & Namioka, A. (eds.). *Participatory design. Principles and practices*, Lawrence Erlbaum, Hillsdale, NJ, 1993, 157-76.
4. Bødker, S., Grønbæk, K. Design in action: From prototyping by demonstration to cooperative prototyping. In: Greenbaum, J.; Kyng, M (eds.). *Design at work: Cooperative design of computer systems*, Lawrence Erlbaum, Hillsdale, NJ, 1991, 197-218.
5. Damsgaard, J. and Scheepers, R. A Stage Model of Intranet Technology Implementation and Management. In: *Proceedings of the 7th European Conference on Information Systems*, 1999, 100–116.
6. Floyd, C. A Systematic Look at Prototyping. In: Budde, R., et al. (eds): *Approaches to Prototyping*. Springer, Berlin, 1984, pp. 1-18.
7. Floyd, C., Reisin, F.-M., Schmidt, G. STEPS to Software Development with Users. In: Ghezzi, C., McDermid, J.A. (eds.). *Proceedings of ESEC '89*, Springer (Lecture Notes 387), Berlin, 1989, pp. 48-64.
8. Jeenicke, M. *Antizipative Anforderungsermittlung bei der Softwareentwicklung*. Master Thesis, University of Hamburg, Department for Informatics, 2001.
9. Jørgensen, N. Putting it all in the trunk: incremental software development in the FreeBSD open source project. *IS Journal 11*, 4 (October 2002).
10. Pape, B., Bleek, W.-G., Jackewitz, I., Janneck, M. Software requirements for project-based learning – CommSy as an exemplary solution. *Proceedings of HICSS-35*, IEEE, 2002
11. Scheepers, R. Key Role Players in the Initiation and Implementation of Intranet Technology. In: *New Information Technologies in Organizational Processes. Proceedings of IFIP WG 8.2*. Chapman and Hall, 1999, 175–195.
12. Sherrell, L. B., Chen, L.-D. The W Life Cycle Model and Associated Methodology for Corporate Web Site Development. *Communications of the Association for Information Systems 5*, Article 7, April 2001.
13. Sommerville, I. *Software Engineering*. 5th edition, Addison-Wesley, Harlow, UK, 1996.
14. Stahl, P., et al. *Analyse und Evaluation der Softwareentwicklung in Deutschland*. GfK Marktforschungs GmbH, 2000, http://www.dlr.de/IT/IV