Bringing Developers and Users closer together: The OpenProposal story

Asarnusch Rashid, Jan Wiesenberger, David Meder, Jan Baumann

FZI Forschungszentrum Informatik
Research Center for Information Technologies at the University of Karlsruhe
Haid-und-Neu Str. 10-14
D-76131 Karlsruhe
rashid@fzi.de, wiesenberger@fzi.de, meder@fzi.de, baumann@fzi.de

Abstract: Even though end-user participation in requirements engineering (RE) is highly important, it is at present not frequently used. Reasons can be found in the large expenditure of time for organizing and carrying out surveys as well as in the time it takes to understand the users' requirements. This research is supposed to address this problem by presenting the OpenProposal approach for distributed user participation using visual requirement specifications. Additionally the research agenda is being outlined for developing and evaluating methods and tools for collaborative end-user participation in Requirements Management (RM). The experiences gathered during this research were promising.

1 Introduction

It is well known that Requirements Engineering (RE) is one of the biggest challenges, and that all stages of RE dealing with elicitation, specification, prioritization and management of requirements are vitally important for the success of a software project. As RE is a complex process involving many people, a lot of different methods and tools were developed to support this highly collaborative process. In the research project CollaBaWü[1] the researchers were confronted with the task to evaluate existing methods and tools for practical usefulness in cooperation with software companies and financial industry and if necessary to develop new methods and tools. In this context, the issue of user involvement in RE has become apparent in many conversations and analyses with the industrial partner.

The aim of user involvement in RE is to improve the design process, facilitate the implementation and to address ethical principles [NJ97]. User involvement can be performed when developing requirement specifications, validating requirement specifications, supporting detailed design and development, reviewing specifications, inspecting prototypes and accepting released products. A literature review [KKL05]

[1] www.collabawue.de: 'CollaBaWü' is a project (2004 – 2007) commissioned by Landesstiftung Baden-Wuerttemberg foundation, which aims at increasing the overall productivity in the development lifecycle of enterprise applications. In this respect, the main objective of the project is to promote industrialisation in enterprise application development focussing on the particularities of the financial service provider domain.

shows, that user involvement is found to have a generally positive effect on system success and on user satisfaction, and some evidence can be found suggesting that taking users as a primary information source is an effective method of requirements elicitation [CM96, EQM96, KC95, Ku03]. In addition, involving users in RE could have an important role, since users are nowadays recognized as being one of the essential groups of stakeholders, while a lack of user involvement was shown to lead to serious problems in RE [Po95]. These results coincide with the experiences of industrial partners (each with more than 10.000 users) in the project CollaBaWü. Their software departments aim at a closer relationship with their users in order to improve their RE processes.

The practitioners already use methods like User Experience, Usability Workshops, User Support Units and already established an employee suggestion system. But recognizing that these procedures are too formal and heavy-weight, and the insight that systematic approaches to understand users' needs and continuous user involvement are still lacking, lead to the suggestion that users have to be able to participate with small effort – the optimum being during their daily work - and developers have to be able to obtain enough valuable information without additional effort. By collaborating, users should exchange and discuss their ideas in a shared environment. Further, transparency is desired, meaning that users can track the development of the suggestions they submitted.

This research aims to contribute to the existing approaches by presenting the OpenProposal system for distributed user participation. The fundamental idea behind OpenProposal is based on the fact that in most modern software products the users' requirements refer directly to the graphical user interface. Therefore the idea of capturing these requirements in a graphical form, supplementing a textual description, was taken into consideration. Since nowadays graphical annotations are unusual in requirements management, and since most of the time sketches and screen shots are merely used on paper, this research wants to discuss the use of graphical annotations in Requirements Management (RM).

For this it is aimed

- to understand the users' and the developers' needs in requirements management,
- to develop a new methodology and a new concept of IT support to enhance existing practices,
- to develop a certain formal notation language providing a language to enable users and software developers to formulate and discuss users' requirements, and finally
- to evaluate the methodology and IT support concept in order to identify chances, risks and limits.

The paper is structured as follows. At first, the theoretical background and related Information System Development (ISD) approaches are discussed. Secondly, the process, concept and architecture of OpenProposal are presented. Then experiences

gained from case studies are outlined. Fourth, possible fields of application of OpenProposal are described and finally, the lessons that could be learned from this research are discussed.

2 Related Work

The OpenProposal concept was able to take its inspiration from several different fields of research, including Participatory Design (PD), Requirements Engineering (RE) and Digital Annotation (DA). The idea for user participation came from the field of PD, as the methods for this particular type of end user integration primarily originate from it [Su93]. PD is solely concerned with user participation in all phases of system development. In general, existing methods for user participation during requirements elicitation utilize direct face to face communication or prefer user interviews. Even though we do not question that these methods allow a complete requirements specification, the successful execution requires considerable effort, both from the side of the developers as well as from the side of the users [DESG00]. Furthermore the increasing degree of interconnectedness through the Internet provides new possibilities of cooperation, but they also imply the need of addressing the challenges of globalization in software development. During the recent years, the employment of CSCW software (Computer Supported Cooperative Work) has therefore been discussed more intensely in PD [KB03]. New approaches in user participation like the concept of Participation Design in Use [SD06] or the new area of Distributed Participatory Design (DPD) [DND06] are supposed to allow the usage of methods of PD in distributed (over time and space) working environments.

In RE research there are a lot of methods and tools available, which support the process of requirements acquisition [HL01]. Besides the traditional concepts of user integration mentioned above, there are many approaches to alternative specification techniques in order to achieve end-user participation. Reeves and Shipman [RS92] describe a combination of visual and textual elements. Li et al. [LDP05] have developed a solution, which uses Natural Language Processing. Video techniques are used in the scenario “Software-Cinema” of Bruegge et al. [Br04], which addresses requirements acquisition in mobile environments. There are also numerous commercial tools supporting visual specification of requirements. The best known notation method is UML, which promotes the use of Use-Case diagrams during the RE phase of software development. These formal techniques support primarily software engineers and presume skills in modelling languages.

Other visual aids in RE include mock-ups [Ma96] and Rapid prototyping techniques [BH98], which are commonly used in early phases of software projects. They do allow software engineers to sketch screens with low functionality as well as to run first usability tests. Tools supporting these techniques are applied by engineers and analysts but not by the user himself. They are merely enabled to return feedback in the usual way of face-to-face meetings. An approach centring more on the user is offered by Moore [Mo03]. It is based on requirements acquisition using GUI elements without functionality. End-users ‘will create mock user interface constructions augmented with

textual argumentation that will act as communication to software requirements engineers' [Mo03, p. 2]. This approach allows users to construct their own user interfaces but still it has to deal with the problem, that real software is very complex and end-users will not have enough time to construct a whole software system.

The advent of graphical user interfaces has led to requirements concerning the modification and improvement of such an interface. Tools supporting this process often use DA. The tool Annotate Pro [An06], for example, provides several functions to draw annotations directly on the users' screen by using snapshots in combination with ordinary picture editing functionality. This enables end-users to draw comments on their active applications, without time-consuming trainings and preparations. The commented snapshots serve as requirement specification and can easily be sent to the software engineers by email. As this tool neither contains a method, which follows a well-structured plan nor provides a formal notation language there does not exist any common language either which means, that users are free to paint sketches and to send them to anyone without assistance. It is assumed that developers may have difficulties in understanding these paintings. Furthermore, there is no possibility for end-users to track the submitted requirements. There are other tools of similar nature, but focussing on different aspects. JING [Ji07] for example, focuses on fast sharing of annotated images and videos. It uses a very basic set of tools, which are nonetheless sufficient for marking and explaining what the central aspects of this picture are. While offering no annotation possibilities for videos, providing a video capture possibility makes the tool flexible. JING also features built in support for upload to Flickr, ScreenCast or any user definable FTP server. However it shares the same shortcomings that Annotate Pro has. Similarly the usability testing suite Morae [Mo07] does not support the structure of the test conducted, nor does it directly support tracking of submitted requirements. It focuses on recording and analyzing usability studies and features a large set of recording options, logging and observation functions as well as tools for analyzing results and creating reports.

The literature about DA states that annotations are not only useful because they allow capturing the application concerned, but they do also support such crucial mental functions such as remembering, clarifying and sharing, as Bottoni et al [BLR03] point out. Their paper provides a formal analysis of digital annotation and identifies operations and procedures that an annotation system should allow. Annotations are also important for a two way information exchange, as discussed by Fogli et al [FFM04], who also define the tools required to support creation and the use of annotations.

In summary, the problem is that none of the present tools and approaches provides all essential functionality to support end-users in an adequate way. All of them are lacking either usability, efficiency or collaboration.

3 OpenProposal: Process, Concept & Implementation

OpenProposal aims at aiding users to express their ideas about how an application might be enhanced. At the same time it is also supposed to help developers by imposing a

structure on the annotation process which will make it easier for them to grasp the users' intention. In order to achieve these requirements, the OpenProposal tool differs from other annotation tools, e.g. Annotate Pro [An06], in so far as the users do not interact with a set of free form drawing tools, but with a toolset representing possible changes he wants his target application to undergo.

OpenProposal is supposed to allow users to annotate their feature requests, error reports or enhancement requests directly on their applications workspace and send these requests to the requirements management. Lots of communication problems can thus be avoided – e.g. misconceptions due to wrong choice of wording, incomplete data, descriptions which are too elaborate – which often arise from text-only communication like E-mail or the internal employee suggestion systems. The aim of OpenProposal is to integrate users efficiently into the development process during their daily routine when using the application, to reduce the usual effort associated with participative requirements elicitation and to allow a high degree of implementation of the captured requirements with the help of structured recording. Furthermore, OpenProposal is supposed to increase the transparency of the requirements management process, thus ensuring motivated participation of as many employees as possible during requirements elicitation.

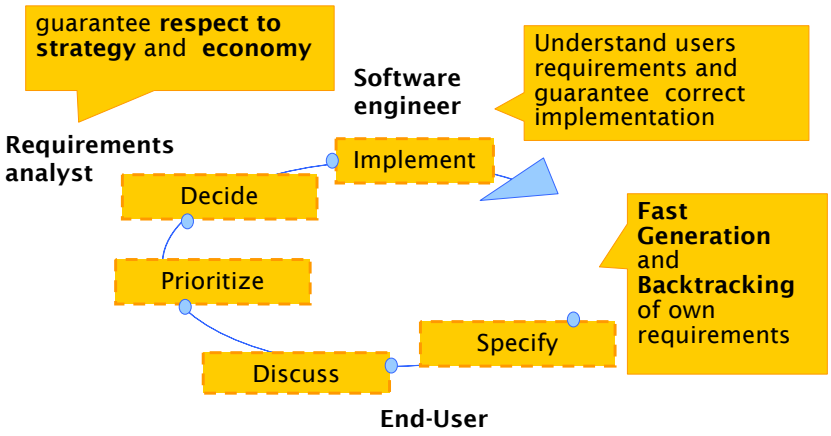

Figure 1: The OpenProposal Process

3.1 Process of OpenProposal

The OpenProposal process (see Figure 1) centers around five actions, specify, discuss, prioritize, decide and implement, and three roles, end user, requirements analyst and software engineer. Each role has its own special set of requirements and participates in a certain subset of the five actions. The end user requires a possibility to generate his own requirements in a fast way. He also wishes to track progress on his requirements. End users take part in the specification and discussion of requirements. The requirements analyst needs to guarantee that whatever is done respects the company's overall strategy and is economically feasible. He takes part in the discussion of the proposals, prioritizes

them and decides on what proposal gets implemented and what doesn't. He may also propose his own ideas and have them discussed with the other stakeholders. The software engineer has as a requirement the need to understand the users' proposals and to guarantee their correct implementation. He can submit his own proposal specifications, participate in discussions to contribute with his professional knowledge of what is technically possible and feasible, and is responsible for implementing the proposals that have been decided on.

3.2 Concept of OpenProposal

To ensure that the involvement of users in distributed RE being successful, a system is needed permitting users to formulate their proposals with simple tools, to submit the created proposal to the developers and to track the proposals progress. Furthermore the decision maker and the software developer should be able to manage and edit the proposals in an efficient way in order to benefit from the possibilities of the RE system, too.

OpenProposal is a software system, which is supposed to fulfil these criteria. It should be possible to use it in conjunction with any established requirements analysis procedure currently employed in the company, which will thus be extended with efficient user involvement. Users should be enabled to create and discuss proposals for existing software as well as software currently being under development and it should be possible to propose improvements as well as new features. The level of detail is up to the user.

The OpenProposal process provides two tools. The annotation tool enables the user to visually formulate his ideas and send them to the collaboration platform tool, which gives an overview of the submitted proposals and allows discussions between users, developers, deciders etc. The process can be initiated in two ways. One way is to explicitly call for user participation, mostly for software which is currently under development. The other way is that the user wants to submit a proposal for an application without external motivation merely wishing to improve the software.

One essential benefit for the user is that he can actively participate in the process of software improvement and is thus able to shape the application the way he wants to. The user employs the annotation tool for the fast generation of graphical requirements and submitting them to the collaboration platform. The user can then track the progress using the collaboration platform.

The system is used by the decision maker to collect and consolidate the users' requirements and to compare them with the strategic and economic targets of the company. If the requirements are collected globally, covering all company divisions, he can detect and use synergies between the divisions. Using the collaboration platform, he can receive an overview of the requirements, can discuss them with users and developers and thus determine the priorities.

The developer benefits from being able to participate at the discussion at an early stage, to inform users, decision makers and analysts about the technical possibilities and restrictions. The graphical specification is supposed to improve the process of understanding what users want, and implement the proposals in the correct way.

3.3 Architecture of OpenProposal

The OpenProposal implementation consists of an annotation tool, an XML specification, an Issue-Tracker and a mediator. As can be seen in Figure 2, the annotation tool gathers annotated screenshots which are stored together with the individual annotation objects in an XML specification. This specification also contains metadata about the proposal as well as the user's system. This specification is sent to a mediator specific to the Issue-Tracker. The mediator takes the information from the specification which will be directly entered into the Issue-Tracker software and creates a new issue with it, attaching the screenshot image and the XML file in the process. Stakeholders can log into the Issue-Tracker to rate and discuss proposals. This information can then be requested by the annotation tool through the mediator from the Issue-Tracker, in order to present users of the annotation tool a list of previous annotations of that application with their ratings and discussions.

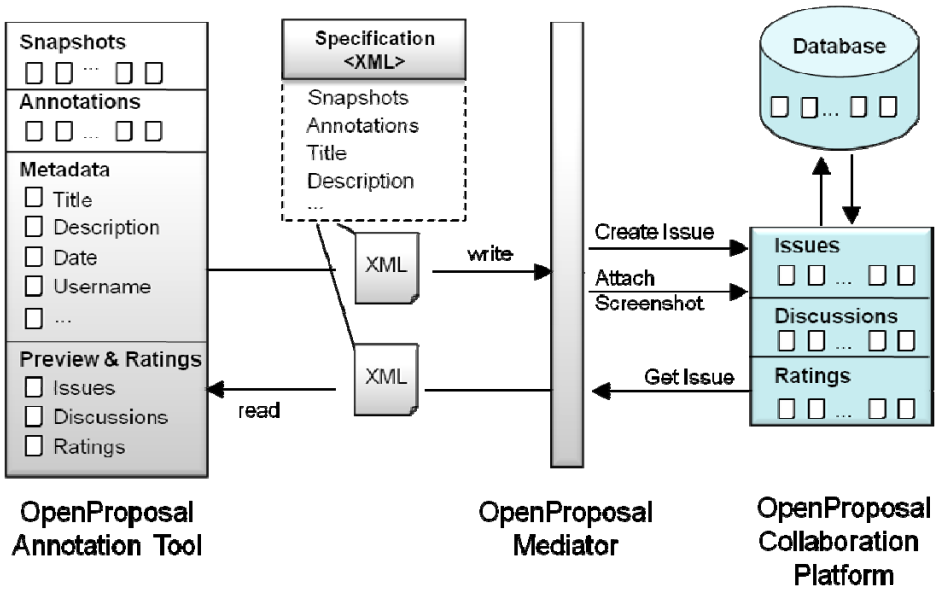

Figure 2: The OpenProposal Architecture

3.3.1 User Interaction: Annotating and Handing-In of Issues

The OpenProposal toolset can be accessed from the tool bar (Figure 3) and offers four annotation related tools. The “Add” tool (1) allows users to specify a position where they

would like to have a new object on the screen. The “Remove” tool (3) is the inverse; an existing element is marked as superfluous. With the “Move” tool (2) users first select an area which should be moved to a different place in the applications workspace, then to the new target area. The “Comment” tool (4) can be used for suggestions the other tools are not able to express directly, as well as refining and adding further detail to the other tools’ annotations. Users may pause the annotation, e.g. if they want to change the layout of the application they are annotating (A).

All annotations are represented as objects which may be edited, moved or deleted whenever the users want to. Once they finish their annotations, users can send their requests to the collaboration platform (H). Prior to sending, the user is prompted to give his request a title and a text description. Users may exit the application at any time.

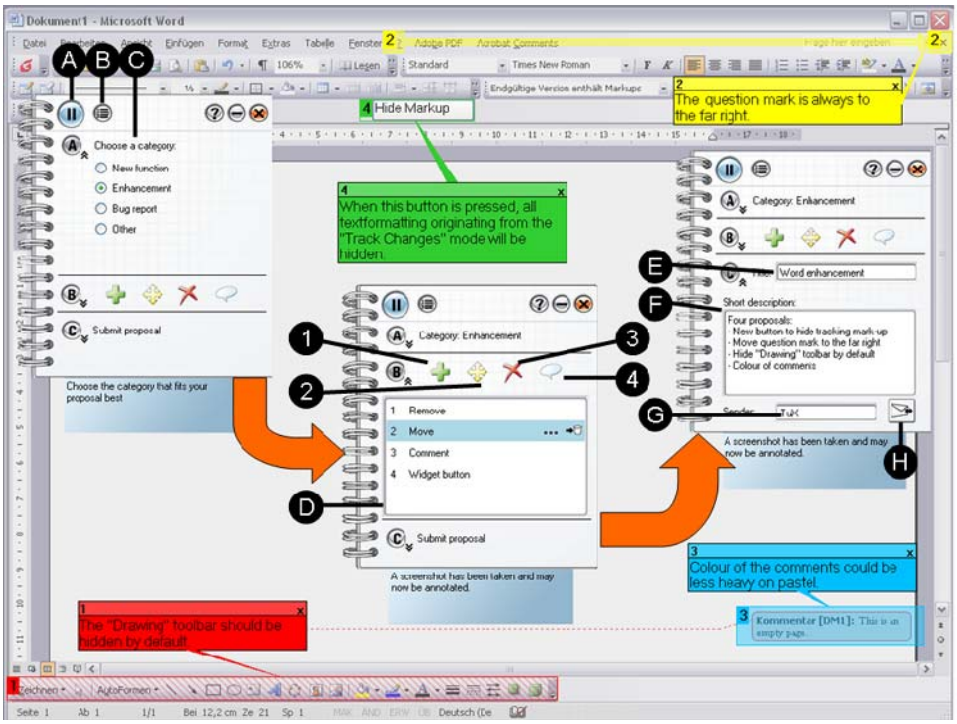

Figure 3: OpenProposal tool bar

3.3.2 Collaboration: Viewing, Discussing and Rating Issues

The data provided by users is stored in the collaboration platforms’ database, which may be accessed via a web front-end. When logging into the collaboration platform, the users are first presented a list of all issues entered. When selecting one of the issues in the list, the issue window is shown (Figure 4). Here the users will be able to read a description of the issue (I), view details such as the status or the priority (L), participate in the

discussion about this issue (K) and take a look at the annotated screenshot associated with this issue (M).

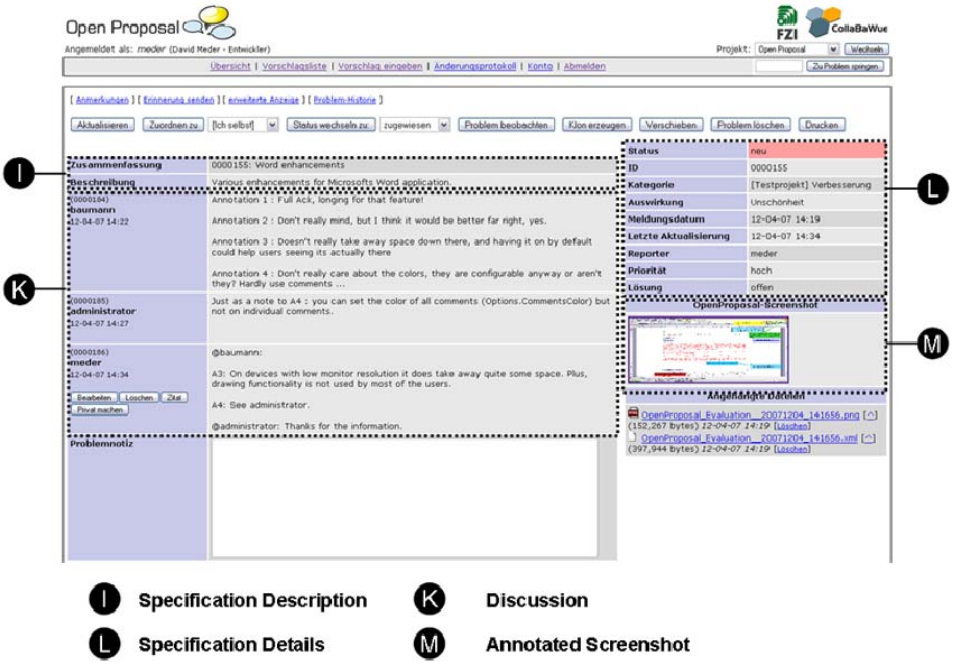

Figure 4: Web-based Issue Tracker

To support the user, the annotation tool of OpenProposal offers a window with a list of all issues (Figure 5) which have been created for the application. This makes it possible to view submitted issues directly in the annotation tool, without the need to open the web-based issue tracker. Because of the large number of specifications, the user is not able to view each specification in the list. In order to reduce the lists' size, OpenProposal provides a filtering function (N). This specification filter works as follows:

1. First all running applications are stored. If a running browser is detected (currently Internet Explorer, Firefox and Opera are supported), the application retrieves the address of the active website.

2. Then the topmost window on the screen is determined. If the user has already created some annotations, the filter also determines the applications which the user has annotated.

3. Lastly, the specifications belonging to one or more applications or website addresses which were determined in the two steps above are shown.

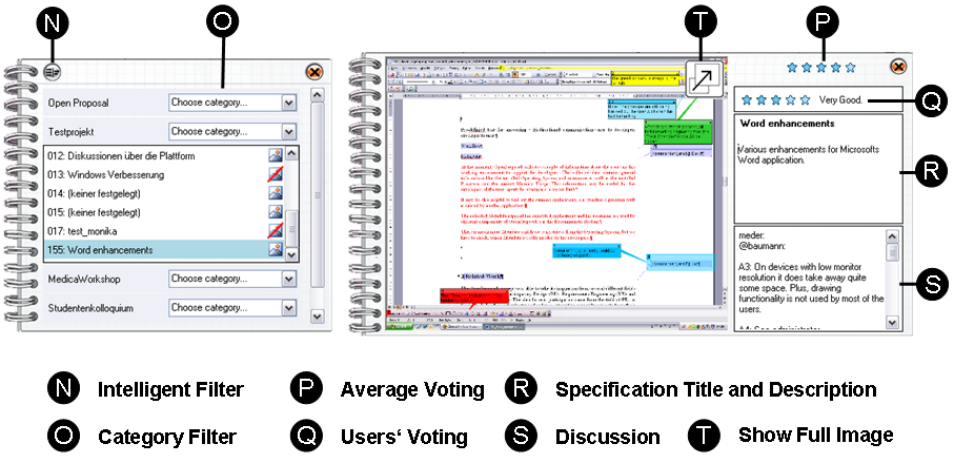

Figure 5: Specification list

The filtered list will be noticeable smaller than the unfiltered list and the user can see through the specifications more easily. Additionally the users can vote for each specification by giving a rating from one to five to express his conformance with the specification.

4 Experiences with OpenProposal

In the course of this research OpenProposal has been evaluated in several steps. First, usability tests were performed to ensure the usability of the annotation tool. Next, case studies with the software development department of the company TRUMPF and a department of the University of Karlsruhe were set up. At present, several case studies in cooperation with small companies specialising in usability design and software development as well as software development departments of larger companies have been started.

4.1 Usability Test of OpenProposal

In February 2007 a first usability and user test was performed with the goal of evaluating the current version regarding usability and user satisfaction. The results of the test would also be used as a basis to create the next version of our OpenProposal application. The 16 test subjects consisted of students from the University of Karlsruhe and employees of the FZI Forschungszentrum Informatik in Karlsruhe, Germany. The test was centred around five annotation tasks the subjects had to perform on an old version of the Firefox[2] browser. A short pre-test interview and a long post-test questionnaire where used to gather information about users thoughts and expectations of the system, as well as the

[2] http://www.mozilla-europe.org/de/products/firefox/

degree to which these expectations where fulfilled. Test subjects where monitored by the investigator the whole time in addition to a video and screen capture being recorded.

The results of the interviews, the observations of the investigator, an analysis of the video recordings and the questionnaires yielded numerous proposals for enhancements, ranging from start-up behaviour (OpenProposal used to switch to annotation mode directly after program start, freezing the users screen in the process – this was later changed due to heavy demand) to interface refinements (the previous version used a separate object list, user demand was an integrated list). But OpenProposal also received encouraging ratings concerning ease of use and usefulness of its annotation concept, for example when being asked if a tool for graphical creation of proposals for software should be provided (averaging to “agree” on a five point scale ranging from “strongly agree” to “strongly disagree”). All in all, the users rated the software as a whole with “good” on a five point scale ranging from “very good” to “very bad”. The enhancement proposals as well as the questionnaire results formed the basic set of changes to be implemented in OpenProposal 2.0.

New possibilities for further studies were found as well. For example, a question during the interview was “What advantages do you think OpenProposal would have?” to which test subjects replied, that it would be faster creating proposals with this application. This assumption could be tested against traditional methods, by taking the time both need for a given task. Some of these questions could be addressed in the subsequent case studies, others required special setups or a long evaluation time and could not be answered yet.

4.2 Case Study ‘TRUMPF’

In September 2007 a second test was done at the TRUMPF company. TRUMPF is a high-tech company focusing on production and medical technology. The TRUMPF Group is one of the world's leading companies in manufacturing technology, with sales of € 1.65 billion/US$ 2 billion and approximately 6500 employees. Since efficient fabrication of high quality components is not a question of hardware alone, TRUMPF also develops the software systems for their hardware, so that the complete process chain – design, programming and production – can be optimized for each other. Usability workshops are a part of the software development cycle at TRUMPF, and such a workshop was used to evaluate our new OpenProposal 2.0 which was used to create proposals for the software being tested during the workshop.

This short case study encompassed 11 test subjects in total and was carried out over two days. There were two groups on each day, one using OpenProposal, the other using another interface for proposals provided by the issue tracking software used by the TRUMPF software development department. The groups were set up so that every test subject would be able to use both interfaces for his proposal, and at the end of each day the subjects were given a questionnaire specific to the interface they used that day. Besides the analysis of the questionnaires’ results, the proposals themselves were evaluated and the developers were interviewed. Additionally, an investigator monitored the study and wrote down comments, problems and observations he made during the workshop.

Question	OpenProposal	Tracker Interface
I was able to quickly find my way in …	Agree	Agree
I often got stuck using … and had to find a work around.	Strongly disagree	Disagree
… shows too much information at once. I found this confusing.	Strongly disagree	Disagree
Proposals are easy to create and don't require much mental effort	Agree	Agree
I made mistakes using …	Disagree	Disagree somewhat
Symbols and naming are easy to understand in …	Agree somewhat	Undecided / Disagree somewhat
The structure of the interface is easy to understand	Agree	Agree
Creating proposals was unnecessarily complex and took a long time.	Strongly Disagree	Disagree

Table 1: Results of the case study 'TRUMPF'

The results showed that OpenProposal was in general well received by the participants. Eight questions on the questionnaire were asked twice, once for OpenProposal and once for the tracker interface (see Table 1). The items were measured on a seven point Likert scale with the options "strongly disagree", "disagree", "disagree somewhat", "undecided", "agree somewhat", "agree", "strongly agree". The first item, "I was able to quickly find my way in …", had an average rating for both systems of "agree". The second item "I often got stuck in … and had to find a work around" received average ratings of "strongly disagree" for OpenProposal, which was thus a bit better than the average of "disagree" for the tracker interface. Similarly the third item "… shows too much information at once. I found this confusing." also received an average of "strongly disagree" for OpenProposal and "disagree" for the tracker. The fourth item, called "Proposals are easy to create and don't require much mental effort", was again rated "agree" for both systems. Item five, "I made mistakes using … ", was rated with "disagree" for OpenProposal and with "disagree somewhat" for the tracker, again OpenProposal received a slightly higher rating. The biggest difference was found at item six "Symbols and naming are easy to understand in … ", where OpenProposal received an average rating of "agree somewhat" and the tracker was rated between "undecided" and "disagree somewhat", another rating where OpenProposal comes out on top. Item seven, "The structure of the interface is easy to understand", received an "agree" rating for both systems and at the last item, "Creating proposals was unnecessarily complex and took a long time.", OpenProposal received another slightly better rating: "strongly disagree" as compared to "disagree". The list of "must fix" enhancement requests was noticeably shorter as well. The participants noted that they did not see OpenProposal as a

replacement of the existing tracker interface, but rather as an easy to use alternative frontend. The software was so well received, that by now it has become an inherent part of the usability process at TRUMPF.

4.3 Case Study 'University of Karlsruhe'

In November 2007 a third usability and user test was launched at the IISM (Institute of Information Systems and Management) at the University of Karlsruhe. The software being tested was a new content management system which would be responsible for the institutes web pages and intranet services. The test phase of the system would be at least two weeks, and users were encouraged to transfer data from the old system to the new during that phase and report any problems or errors they found in the process. To ease the reporting process, OpenProposal would be provided to all users and would be configured so proposals are directly sent to the issue tracker included in the content management system.

In a first step, the first usability and user test was replicated with several test subjects from the IISM. This was done to evaluate, if the new version was indeed an improvement over the old version. The second step began with the introduction of the new content management system as well as OpenProposal in mid November. A time frame of two weeks was given to the participants to get used to the new system, file bug reports, enhancement and feature requests and transfer data. This phase of the test is still in progress.

Test Item	Median Feb07	Median Nov07	p
Handling the user interface was [easy, medium, hard]	medium	easy	4,67 %
There are [none, some, many] functions I miss in OpenProposal	some	none	0,50 %
OpenProposal sufficiently informs me about what it is doing at the moment [strongly agree, agree, neutral, disagree, strongly disagree]	neutral	strongly agree	2,38 %
OpenProposal has a persistent style of handling throughout the whole program [strongly agree, agree, neutral, disagree, strongly disagree]	agree	strongly agree	3,71 %
Technical performance of OpenProposal was [very good, good, neutral, bad, very bad]	neutral	very good	1,52 %

Table 2: Results of the Case Study 'University of Karlsruhe'

Only results of the replication test are available as of now. Due to time constraints only five test subjects could be interviewed and observed. A Mann-Whitney-U test was performed for each question on the two sets of answers (one from February, one from

November). This statistical non-parametric test can be used to check if two samples have equal distributions, in this case meaning that if the test yields a significant result, the sets of answers can be considered statistically different. According to Albers et al [AKK07] the level of significance is usually set to 5% for significant results and 1% for highly significant results. For five of the 30 items the Mann-Whitney-U test calculated a significant difference ($p<5\%$), all other differences in sets of answers were not significant (see Table 2). The first statistically different item was “Handling the user interface was … “ with the three choices “easy”, “medium” and “hard” being possible. While in February participants on average answered with “medium”, in November the average answer was “easy”, the difference being statistically different with a significance of $p=4{,}67\%$. The second item was “There are … functions I miss in OpenProposal” with the three options “none”, “some”, “many” where the first test in February had an average result of “some” while the second test in November had an average result of “none”, the difference being highly significant with a significance of $p=0{,}50\%$. The third item was “OpenProposal sufficiently informs me about what it is doing at the moment” with the five options “strongly agree”, “agree”, “neutral”, “disagree” and “strongly disagree”. The average answer in February was “neutral”, while in November the participants on average choose “strongly agree”, the significance being $p=2{,}38\%$. When being asked about persistency of handling, “OpenProposal has a persistent style of handling throughout the whole program” with the options “strongly agree”, “agree”, “neutral”, “disagree”, “strongly disagree”, the participants of the first test in February answered on average with “agree” while in November the average answer was “strongly agree”. The difference had a significance of $p=3{,}71\%$. Lastly, the item “Technical performance of OpenProposal was …”, with the choices ranging from “very good”, “good”, “neutral”, “bad”, “very bad”, was rated with an average of “neutral” in the first test, while the November test had an average rating of “very good”, the significance of this difference being $p=1{,}52\%$. This shows that in all significantly different results, the new version received better ratings than the old version and can thus be considered an improvement.

5 Possible Fields of Application of OpenProposal

OpenProposal is supposed to support the RE phase of the software development process. During the course of the research project, it became apparent that the most promising fields seemed to be ‘Usability Tests in Usability Workshops’, ‘Support and Maintenance of Software’ and ‘Global Software Development’.

5.1 Usability Test of Software in Usability Workshops

OpenProposal was built with a non-intrusive integration into the users’ workflow in mind. A field of application where this is especially helpful is that of usability workshops. The purpose of these workshops is an evaluation of an existing piece of software using people which correspond to the actual user as well as possible. The most important sources of information in these workshops are the test subjects themselves, especially their suggestions and ideas for improving the software at hand. Traditionally, when making such a suggestion the test subject would either make a handwritten note on

a piece of paper, switch application context to write an electronic note or report their suggestion to the investigator. The best traditional way would be the third option, since it allows questions for clarification and refinement of the new proposal. This however would require having close to as many investigators as there are test subjects, making this option expensive. Having the test subject switch application context makes writing proposals cumbersome for the subject, since he may need to switch back and forth a number of times to write a good proposal. This can be solved when using handwritten notes. Here however the problem arises, that these notes need to be deciphered and converted into an electronic format. While OpenProposal cannot replace an investigator, when time and money are an issue it is likely to perform better than both alternative methods. The user does not have to switch context to some other application, but creates his annotation directly inside his current context. And there are no handwritten notes which need to be processed after the workshop improving clearness and correctness of the resulting proposals.

5.2 Support and Maintenance of Software

When considering software support, OpenProposal allows the end user to graphically formulate his problem – if it is referring to the user interface - and send it to support, for example via e-mail. The support team can then quickly detect the user's problem without reading a long textual description of the problem. On the other hand OpenProposal can also be used by the support team, helping to translate the user's support request into a graphical specification and sending it onto an Issue-Tracking platform. Similarly, OpenProposal can be used for software maintenance, where both end users as well as developers can file bug/error reports and improvement proposals as well as discuss the existing proposals.

5.3 Global Software Development

Modern software products often tend to be highly complex. Their development and production requires a lot of expertise and competence that can rarely be found in one place, at least not with the economy necessary for a fiercely competitive, global world. Thus, modern software production tends to be highly fragmented, geographically distributed on a more or less global scale, where each participant is specialized in its own core competence. The resulting "global" software products should have the same quality at a more attractive price to those one would achieve at a single place. The design, the development and the production processes for a global software development involve new competencies in communication, collaboration, integration, and technical and managerial control.

There are quiet some problems in global software development, particular to communication or information sharing. Especial problems like time zone difference or geographical distance hinder successful communication. Nowadays these problems are reduced by the wide avaibility of modern communication techniques like the internet.

OpenProposal supports the requirements engineering in geographically distributed environments. The tool supports the communication between the project team, users and customers occur in a geographically distributed way using modern communication techniques to reduce the impact of the geographical dispersion, e.g. different time zones. By generating requirements descriptions it supports the user in a formal and understandable way.

6 Summary & Outlook

In this research, the concept and first evaluation of OpenProposal are presented. All the conclusions drawn so far are based on software companies' practical knowledge as well as on previous related research and the results of the first case studies. The results of the first usability tests and user surveys are promising, but significant results can only be delivered by the end of the case-study. As the case study is the most essential part of this research, the methodology for the evaluation has to be thoroughly chosen and performed. It is crucially important to identify success factors and risks in order to gain a better understanding about collaborative end-user participation in requirements management and to provide adequate solutions for IT support.

The research described above can help to improve the communication processes in software development. Primary purpose of the research is to show the practicability of end-user participation and to establish a new method in requirements management. It seems possible that this work can reveal new findings about the way users and software developers interact and can therefore offer new opportunities for innovative ways of collaboration in RE e.g. corresponding methods in Global Software Development.

Connections between employee satisfaction with regard to the direct influence on the design of their own used software, and the acceptance of the above described annotation tool are the decisive factor for further developments.

In particular, case studies have to point out how much actual effort will have to be spent by using the system described above and in which amount requirements will have to be specified. The degree of quality in the end-users' specification is another important key figure, which defines the applicability of the submitted requirements and determines whether the users' specification are effectively helpful. Further, it is not yet clear how far end-users have to be supported by specification templates without restricting the users' creativity and, simultaneously, understanding the users' drafts with a reasonable effort.

There are several challenges when trying to integrate such a requirements acquisition system in an organisation. Motivating end-users to take part is of vital importance. Further, any formalities like data privacy have to be complied with. For example, if the annotation tool is used in an application which contains private data, it must be clear how to handle the annotated screenshots. Furthermore, software engineers and requirements analysts have to familiarize themselves with responding to end-user requirements more often. The integration of specifications annotated by the end-users

into software engineers' methods and tools is another challenge. For example, the enhancement of UML could be useful.

On the whole, all depends on the acceptance and efficiency of the integration of such a system. By 'all-time' acquiring of end-users requirements organisations can make themselves more flexible and improve their long-period planning, as future tendencies of end-users needs are recognized in a shorter time. It should be investigated, if GUI-dependent and workflow concerning requirements could be specified structured by end-users with easy usable tools. In the center of interest are audio-visual tools as well as the effort of assistants, which should supply the end-user by modelling his requirements.

References

[AKK07] Albers, S.; Klapper, D.; Konradt, U.; Walter, A.; Wolf, J.: Methodik der empirischen Forschung, Gabler Verlag, Wiesbaden, Germany, 2007.

[An06] Annotate Pro, http://www.annotatepro.com/, viewed on 13.06.2006.

[BH98] Beynon-Davies, P.; Holmes, S.: Integrating rapid application development and participatory design, In: IEEE Software, Volume 145, Issue 4, pp. 105-112, 1998.

[BLR03] Bottoni, P.; Levialdi, S.; Rizzo, P.: An Analysis and Case Study of Digital Annotation. In: Bianchi-Berthouze, N. (Eds.): Proc. 3rd International Workshop on Databases in Networked Information Systems, Aizu-Wakamatsu, Japan, 2003, pp. 216 - 230. Lecture Notes in Computer Science 2822, Springer, Heidelberg, Germany, 2003.

[Br04] Bruegge B.; Creighton, O.; Purvis, M.: Software Cinema, CHI Workshop on Identifying Gaps between HCI, Software Engineering and Design, and Boundary Objects to Bridge Them, Vienna, Austria, 2004.

[CM96] Chatzoglou, P.C.; Macaulay, L.: Requirements Capture and Analysis: A Survey of Current Practice, Requirements Engineering, Volume 1, Issue 2, pp. 75-87, 1996.

[DESG00] Damian, D.E. H.; Eberlein, A.; Shaw, M.L.G.; Gaines, B.R.: Using different communication media in requirements negotiation, IEEE Software Volume 17, Issue 3, May-June, pp. 28-36, 2000.

[DND06] Danielson, K., Naghsh, A.M., Dearden, A.: Distributed Participatory Design. Extended Abstract of the workshop for Distributed Participatory Design at conference NordiCHI' 06, 2006.

[EQM96] El Emam, K.; Quintin, S.; Madhavji, N.H.: User Participation in the Requirements Engineering Process: An Empirical Study, Requirements Engineering, Volume 1, Issue 1, pp. 4-26, 1996.

[FFM04] Fogli, D.; Fresta, G.; Mussio, P.: On Electronic Annotation and its Implementation. In: Proceedings of the working conference on advanced visual interfaces, Gallipoli, Italy, pp. 98 – 102, 2004.

[HL01] Hofmann, H.F.; Lehner, F.: Requirements: Engineering as a Success Factor in Sofware Projects, IEEE Software Volume 19, Issue 4, Regensburg, Germany, pp. 58-66, 2001.

[Ji07] JING, http://www.jingproject.com/, viewed on 03.12.2007.

[KB03] Kensing, F.; Blomberg, J: Participatory Design: Issues and Concerns. In: Kensing, F.: Methods and Practices in Participatory Design. Copenhagen. ITU Press Copenhagen, Denmark, pp. 365-387, 2003.

[KC95] Keil, M.; Carmel, E.: Customer-Developer Links in Software Development, Communications if the ACM, Volume 38, Issue 5, pp. 43-51, 1995.

[KKL05] Kujala, S.; Kauppinnen, M.; Lehtola, L.; Kojo, T.: The Role of User Involvement in Requirements Quality and Project Success, IEEE International Conference on Requirements Engineering (RE'05), 2005.

[Ku05] Kujala, S.: User Involvement: A Review of Benefits and Challenges, Behavior & Information Technology, Volume 22, Issue 1, pp. 1-16, 2003.

[LDP05] Li, K.; Dewar, R.G.; Pooley, R.J.: Computer-Assisted and Customer Oriented Requirements Elicitation, Proceedings of the 13th IEEE International Conference on Requirements Engineering, Edinburgh, UK, 2005, pp. 479- 480, 2005.

[Ma96] Macaulay, L.: Requirements for Requirements Engineering Technique, Second International Conference on Requirements Engineering, (ICRE'96), Colorado Springs, USA, p. 157, 1996.

[Mo03] Moore, J.M.: Communicating Requirements Using End-User GUI Constructions with Argumentation, Proceedings.of the 18th IEEE International Conference on Automated Software Engineering ASE'03, Montral, Canada, pp. 360 – 363, 2003.

[Mo07] MORAE, http://www.techsmith.com/morae.asp, viewed on 03.12.2007

[NJ97] Nandhakumar, J.; Jones, M.: Designing in the Dark: the Changing User-Developer Relationship in Information Systems Development, Proc. ICIS, 1997.

[Po95] Potts, C.: Software Engineering Research Revisited, IEEE Software, Volume 10, Issue 5, pp. 19-28, 1995.

[RS92] Reeves, B.; Shipman, F.: Supporting Communication between Designers with Atrifact-Centred Evolving Information Spaces, Proceedings of the CSCQ '92, Toronto, Canada, pp. 394-401, 1992.

[SD06] Stevens, G.; Draxler, S.: Partizipation im Nutzungskontext. In: Heinecke, A.M.; Paul, H. (Eds.): Mensch & Computer 2006. Oldenbourg Verlag, Munic, Gemany, pp. 83- 92, 2006.

[Su93] Suchmann, L.: Forword. In: Schuler, D.; Namioka, A. (Hrsg.): Participatory Design: Principles and Practices. Lawrence Erlbaum, New York, pp. vii – ix, 1993.