Department of Computer Science
Aalborg University

TITLE:

Usability Evaluation in Open Source Development

SEMESTER:

INF 8,
From 1st of February 2006
to 8th of June 2006

PROJECT GROUP:

E3-211

Authors:

Morten Sieker Andreasen

Henrik Villemann Nielsen

Simon Ormholt Schrøder

SUPERVISOR:

Jan Stage

ABSTRACT:

Open Source Software has become an increasingly popular alternative to commercial software but it has been criticized for it's lack of usability. This master thesis shows how usability work in Open Source Software development can be improved using remote usability evaluation.

Initially a questionnaire survey and interviews with open source developers and usability professionals were performed in order to investigate the opinions towards usability and the current practice of usability evaluation. The importance of usability was regarded as high among open source developers but in practice the main usability effort was based on common sense. The main obstacles for improved usability work were found in the open source development model and in the distributed organizational structure.

Therefore a comparative study of three remote usability evaluation methods was performed against a laboratory evaluation. This comparison showed that the results of a remote synchronous evaluation were comparable to those of a laboratory evaluation. An asynchronous remote evaluation identified significantly fewer problems than a laboratory evaluation but the problems identified were critical.

We consider both methods alternatives or supplements for usability evaluation in the open source community.

Summary

This master thesis treats the overall topics: Usability evaluation and Open Source Software (OSS) development. We have investigated these topics in two studies. In one study we conducted three surveys in order to explore how contributors working within OSS considered usability. In the second study we completed a methodical comparison of three approaches for remote usability evaluation to a laboratory evaluation.

OSS is a concept that covers software which is released under a license that allows the end user to freely use, distribute and modify the source code of the program. Famous OSS projects include Linux, Eclipse, Apache, Tomcat and the Mozilla suite of programs. The majority of OSS projects are developed by individuals or small development teams with little formal organization. Though the OSS community has produced software products able to compete with commercial alternatives, OSS development also faces a number of fundamental challenges. The premises of OSS are often vastly different from traditional software development environments; developers often work in their spare time and most communication needs to be by electronic means since developers often are distributed around the world.

In recent years there has been increasing attention towards the technological achievements of OSS. At the same time OSS has been criticized for it's lack of usability for non-technical users compared to commercial software. In this context we wanted to investigate the opinions and practice towards usability within the OSS community. We did this by conducting a questionnaire survey which was sent out to fourteen OSS projects. Following this we conducted three interviews with OSS developers where we investigated the findings from the questionnaire further. In order to reduce bias we also conducted a focus group interview and personal interviews with five usability professionals contributing to OSS.

The first study concluded that even though OSS contributors were interested in usability and regarded the importance as high in their projects, in practice the usability effort was limited. Most usability considerations were based on common sense, design guidelines and conventions from other programs. We found that conventional usability evaluation methods did not support the OSS development paradigm and we wanted to investigate if alternatives existed that solved the following obstacles: Lack of access to usability laboratories, lack of financial resources, the

geographically distributed nature and short development cycles.

We wanted to explore whether remote usability evaluation could help to overcome the obstacles in the OSS community. Remote usability evaluation is a term used when the evaluators are separated from users in space and possibly time. When evaluating using synchronous methods the evaluator is separated from the user in space but not in time. On the other hand when conducting an asynchronous evaluation the evaluator is separated from the user in time and possibly in space.

We conducted a comparative study in which we compared three approaches for remote usability evaluation to a conventional laboratory evaluation. The following methods were compared:

- Laboratory evaluation (LAB)
- Remote synchronous evaluation (RS)
- Asynchronous expert evaluation (AE)
- Asynchronous user evaluation (AU)

The comparison revealed a number of findings which supported the idea of introducing remote usability methodologies to the OSS community. The study concluded that the results of an RS evaluation were comparable to those of a LAB evaluation. They identified almost the same number of usability problems. In addition, test users spent the same time completing the evaluation. The AE and AU evaluations identified fewer problems than the LAB evaluation. However, almost all reported problems turned out to be critical. There was no significant difference between the number of problems identified by the usability experts and the ordinary users.

The overall conclusion was that the remote methods have the potential to solve some of the obstacles to usability work in the OSS community.

Preface

This master thesis deals with usability evaluation in relation to Open Source Software (OSS) development. It consists of this report and two individual research papers. The work would not have been possible without the invaluable assistance from a number of people.

First and foremost we want to thank our project supervisor Jan Stage for guiding us and providing constructive feedback on methodical as well as structural difficulties. Furthermore, we want to thank Mikael Skov for reviewing our methodology concerning the data analysis of the performed evaluations. Mikael also provided us with insight in statistical analysis.

In regard to the interviews of this study we would like to thank Anders Rune Jensen, 'Taupter' and Duncan, and the employees at Relevantive. We especially thank Ellen Reitmayr and Björn Balazs for taking extra time to answer our questions. We also performed twenty-four usability evaluations and in this connection, we thank the participants for volunteering their time. Finally, we thank the respondents of the questionnaire.

Bibliographical Reference Format

When referring to literature we use square brackets [x] where x denotes the unique number given to each item in the bibliography.

Contents

CHAPTER 1

Introduction

Attention for OSS has increased in recent years and it is no longer just used by computer experts. However, usability problems in many OSS user interfaces are well documented [6, 11, 14, 28]. The premises of OSS are different from most commercial software projects; for instance OSS is developed in distributed collaboration between potentially large number of programmers and most projects do not receive any funding.

1.1 Research Questions

The two main themes covered in this master thesis are OSS and usability. More specific we have investigated the opinions and practice towards usability in the open source community and made a comparison of three approaches to remote usability evaluation.

The overall research question in this thesis is:

> *How can the usability work in OSS development be improved?*

In order to answer this question we present the two research questions which have guided the course of our research.

1.1.1 Research Question 1

Little research has been conducted concerning reasons that OSS generally does not have the same degree of usability as commercial software. If the state of affairs regarding usability is to change, the obstacles need to be identified. The first research question of the thesis is:

> *How is usability currently considered by both OSS developers and usability experts participating in OSS projects, what is the current practice of usability evaluation within OSS development, and what are the obstacles for change?*

1.1.2 Research Question 2

The area of remote usability evaluation is relatively new and no methodical comparison have been made of multiple types of remote usability evaluations. The second research question of the thesis is:

> *How do three different approaches to remote usability evaluation compare to a laboratory evaluation in regard to the identification of usability problems?*

CHAPTER 2

Open Source Development

Today most people involved with software development and the IT industry have heard of the term OSS. In the following we present an overview of the history of OSS and present and discuss the OSS development methodology.

2.1 History

OSS is a concept that has evolved over the last 25 years. In the early days of computing most software was distributed freely among academics and professionals but by the 1980's software was largely sold as a commodity with strict limitations [13, 20, 31]. In 1984 the Free Software Foundation (FSF) came to existence originating as a protest to the rising commercialization of the software community. The FSF insists on using the term 'Free Software' referring to "*Free as in speech, not free as in beer*" [36]. The FSF is a highly ideological organization and it developed the GNU[1] General Public License (GPL) [35], under which a large share of Free Software is distributed. FSF's founder Richard Stallman describes proprietary software as a social problem that limits the freedom of the users and he lists four basic liberties that software needs to comply to in order to be labeled Free Software [37]:

- The freedom to run the program, for any purpose.
- The freedom to study how the program works, and adapt it to your needs. Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbour.
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. Access to the source code is a precondition for this.

[1] Acronym for "Gnu's Not UNIX". A UNIX-compatible operating system developed by the Free Software Foundation and the basis of the General Public License (GPL)

Within the technological community and the IT industry opinions vary on the benefits of these freedoms. As one of the leading commercial software companies, Microsoft has repeatedly criticized those sharing their software for undermining the foundation of the software industry. Already in 1976 Bill Gates sent out an open letter to hobbyists stating that they were "*stealing*" his software by sharing it [15]. Today Microsoft is one of the prominent advocates of copyrights and patents in the software world, and this is the foundation upon which they evolved into the world's largest distributor of software with 61.000 employees worldwide and an annual turnover exceeding 40 billion dollars in 2005.

In 1991 the Finnish computer science student Linus Torvalds started developing the Linux operating system under GPL and encouraged other hackers[2] to contribute to the project. During the 1990's and the rise of the internet, Linux became increasingly popular and commercial companies started to turn their attention to Linux and other Free Software projects. In 1998 Netscape was loosing the 'browser war' against Microsoft's Internet Explorer and decided to turn it's Netscape Navigator into an OSS product. The resulting Mozilla Foundation revived the browser and attracted contributors from the OSS community [39]. Around the same time the Open Source Initiative (OSI) was founded by Eric Raymond and Bruce Perens. They were concerned that businesses did not find Stallman's freedom rhetoric appealing and wanted to form an organization that attracted the commercial world. OSI was less ideological and had more pragmatic views than the FSF and chose the term 'Open Source Software' instead of 'Free Software'. There were principal differences between OSI and FSF but they largely agreed on the main principles and goals of the effort against proprietary software [31, 36]. Today OSI is a non-profit corporation dedicated to managing and promoting the 'Open Source Definition' and maintaining a variety of open source licenses such as GPL and BSD[3] (Table 2.1). The main difference between the GPL and BSD is that derivative work of software under a BSD license does not need to be OSS. Although the majority of OSS is developed by individuals and small development teams, many corporations also distribute software under one or more of these licenses. For instance IBM and Novell are deeply involved with OSS development and the kernel of Apple's OSX operating system is based on a BSD license.

To many OSS contributors proprietary software is considered the main opponent and the commercial world is increasingly aware of OSS. In 1998 a number of internal Microsoft memos, the so called 'Halloween documents', were leaked onto the internet. The documents revealed that "*OSS poses a direct, short-term revenue and platform threat to Microsoft, particularly in server space. Additionally, the intrinsic parallelism and free idea exchange in OSS has benefits that are not replicable with our current licensing model and therefore present a long term developer mind share threat*" [40]. Microsoft has repeatedly defended the commercial business model with emphasis on intellectual property as the only serious alternative, when it comes to financially viable software development. They have discouraged consumers to turn to OSS and Linux by associating it to the failed projects of the 'dot-com bubble' of the late 1990's [27]. Though companies like IBM do make substantial profits on Linux and OSS, it is true that many OSS companies are struggling generating profits [30].

[2] A slang term for a computer enthusiast. Not to be confused with the term 'cracker' that refers to a person committing criminal acts online.

[3] Berkeley Software Distribution: A free license developed by the University of California at Berkeley.

	Non-free software					OSS	
Attribute	Commercial	Shared Source	Trial	Shareware	Freeware	BSD	GPL
Zero initial cost			(x)	(x)	x	x	x
Redistributable		(x)	x	x	x	x	x
Unlimited usage					x	x	x
Source code available		(x)				x	x
Source code modifiable						x	x
All derivatives must be free							x

Table 2.1: An overview of various software licenses or ways of distribution [40]. **Notes:** Usually trial software and shareware is only available at no cost in limited editions. Shared Source is a license type launched by Microsoft that gives business partners access to source code.

As a response to OSS Microsoft released the 'Shared Source' license in 2001 [27], which gave limited access to the source code of a selection of Microsoft's programs to selected customers and partners, in order to counter increasing requests for openness of Microsoft products (Table 2.1).

Some parts of the debate about OSS can be characterized as mud slinging from both sides. The CEO of Microsoft Steve Ballmer stated that "*Linux is a cancer that attaches itself in an intellectual property sense to everything it touches*" in an interview with the Chicago Sun-Times [2], referring to the fact that the GPL license requires all derivatives to be OSS as well. This issue has been one of the major themes in the OSS debate. In relation to this the company SCO, which claims to hold the rights to the UNIX operating system, has sued companies involved with OSS such as IBM, Red Hat and Novell. The CEO of SCO Darl McBride has furthermore claimed that GPL violates the American Constitution, is a direct attack against international copyright laws, and even poses a threat to American national security, since it allows rogue nations to obtain advanced technology [24, 25]. OSS proponents have answered back using the same level of shrill rhetoric and have compared the Shared Source license to a virus: "*Shared source, therefore, behaves like a virus that infects developers' brains. Once you let it into your organization, you must keep careful track of which developers have been contaminated, avoid deploying them to any projects which might compete with a Microsoft product, and even erect "Chinese walls" between projects so that no knowledge from shared source can leak into projects with competitive implications.*" [29]. They also claim that the patent based business model threatens the software industry as a whole.

2.2 The OSS Development Process

The principle behind OSS development is simple. When the source code for a piece of software can be freely read, redistributed, and modified, the software evolves. This way development and project management in OSS varies from most commercial software projects. In the famous paper 'The Cathedral and the Bazaar' Eric Raymond described the OSS development process with the metaphor of "*a great babbling bazaar of differing agendas and approaches*" [32]. In his opinion proprietary software development is comparable to cathedral building with long release cycles and no openness in the development process. Linus Torvalds compared OSS development to the world of science: "*science took this whole notion of developing ideas in the open and improving on other peoples' ideas and making it into what science is today*" and furthermore stated that proprietary software resembled witchcraft and alchemy, because of it's secrecy [9]. A practical argument for openness that OSS developers often use is "*Given enough eyeballs, all bugs are shallow*" [32]. Mockus et al [26] described the main characteristics of OSS development as:

- OSS systems are built by potentially large numbers of volunteers
- Work is not assigned; people undertake the work they choose to undertake
- There is no explicit system level design, or even detailed design
- There is no project plan, schedule, or list of deliverables

The possibility of an OSS project splitting up binds the projects together. Karl Fogel described the social and political infrastructure of OSS, and listed a number of factors that have turned out to be essential for successful OSS projects in regard to technical quality, operational health and survivability [13]. One of the important attributes of OSS was the possibility of forking. This means that the project can be split up, if contributors disagree about the way the project is heading. Fogel claims that this is the main ingredient that binds developers together on OSS projects. "*The paradoxical thing is that the possibility of forks is usually a much greater force in free software projects than actual forks, which are very rare*". Though it is common to refer to OSS project managers as 'benevolent dictators', the fork-ability of OSS makes it possible for contributors to continue a fork of the project without the current leader, if the 'tyrant' looses the trust of his subjects. Therefore OSS leaders often let things work themselves out through discussion and only make decisions when no consensus can be reached [13]. When OSS projects mature they tend to move away from 'dictatorship' and establish the foundation of more democratic systems.

A good, motivating working environment is vital in OSS development. The community around an OSS project is considered the main factor that results in quality software. For instance Brian Behlendof of the Apache Software Foundation stated "*The Apache Software Foundation, for example, believes that it's first order of business is creating healthy software developer communities focused on solving common problems: good software is simply an emergent result*" [13]. Beside abilities like good structure design skills and communicative talent, one of the most important roles of an OSS leader is to nurture the developer community and make others willing to join the project and continue to make contributions [13]. Hence, a good OSS leader should concentrate his efforts on making sure that contributors feel appreciated and that there is a positive working environment.

OSS emphasizes other stages of development than traditional software engineering. Paul Vixie compared OSS development to the following seven stages of traditional software engineering [41]:

1. Marketing requirements
2. System level design
3. Detailed design
4. Implementation
5. Integration
6. Field testing
7. Support

Vixie stated that the Marketing requirements are not a high priority in OSS since "*Open Source folks tend to build the tools they need or wish they had*". Often OSS projects are initiated in order to scratch a developer's personal 'itch' [32]. He also found that unfunded OSS projects initially do not have System level design. The overall design of the system evolves during the first couple of versions of the program. The Detailed design is also not a high priority to OSS developers and programming interfaces are often not documented very well. The stage of Implementation is considered the fun part: "*The opportunity to write code is the primary motivation for almost all open-source software development effort ever expended*". In OSS the Integration stage usually means making sure that the software builds on every kind of system the developer has access to and making a ready-to-download package for users. There is little pre-release testing and usually no unit tests, regression or otherwise. However, Vixen states that the post-release stage of Field testing is the major strength of OSS: "*The essence of field testing is its lack of rigor. What software engineering is looking for from its field testers is patterns of use which are inherently unpredictable at the time the system is being designed and built–in other words, real world experiences of real users. Unfunded open-source projects are simply unbeatable in this area*". There is little focus on the Support stage within OSS and usually the developers themselves do not provide any; "*'Oops, sorry!' is what's usually said when a user finds a bug, or 'Oops, sorry, and thanks!' if they also send a patch*". Support can be maintained through the community or consultants specialized in OSS.
Overall the 'OSS model' can therefore be considered a development framework rather than a step-by-step development model. To get the benefits of the OSS framework Fogel recommended going through the following steps when initiating an OSS project [13]:

- Look around: Are there any existing projects doing what you need?
- Build infrastructure: Set up a mailing list, a website, version control system, bug tracking system and real time chat.
- Mission statement: Describe the goal of the project.
- Developer guidelines: Make guides available to new contributors.
- Documentation: As a minimum there should be a description of how to set up the software and perform basic diagnostic tests in case it does not work.
- License: Choose an OSS license and apply it.
- Set the tone of the community: Make sure that there is a good working environment, avoid private discussions and make sure that rudeness is not accepted in discussions.

- Code reviews: Setup systems for code reviews to ensure quality.

Many have strong opinions about the strengths and weaknesses of OSS and the development method, but little formal research has been performed to investigate the claims. We have listed the main positive and negative attributes of OSS that have been described by Mockus et al, Feller et al and Vixie [12, 26, 41] (Table 2.2).

Advantages	Disadvantages
• Speed of development • Access to code and lack of 'black boxes' is important for code inspections to assure dependability and correctness • Peer reviews result in better code • Large pool of testers and developers facilitates debugging • Supports geographically distributed development teams	• No formal System Level Design procedures • No formal specification of requirements • Developer centered user interfaces • Lack of documentation • Lack of support and warranty • Potential intellectual property problems

Table 2.2: The table summarizes the advantages and disadvantages of the OSS development model.

Overall OSS or Free Software development was founded in ideology and has evolved into a commonly used approach to software development. Opinions about OSS are divided and there are indeed both strengths and weaknesses of the approach. The 'OSS development model' can be considered a development framework rather than a full scale software engineering strategy. Compared to commercial software engineering, OSS development places strong emphasis on the stages of implementation and field testing and often disregards the design stages. This approach does have advantages like the possibility of rapid software development and a thoroughly tested code base but from a customer perspective, the lack of a formal design stage, documentation and support can seem like major disadvantages.

CHAPTER 3

Research Papers

This chapter presents the two research papers of the thesis, which are included in the appendices (Paper 1: Appendix A on page 33 and Paper 2: Appendix B on page 47). The relation between the research papers is illustrated in Figure 3.1.

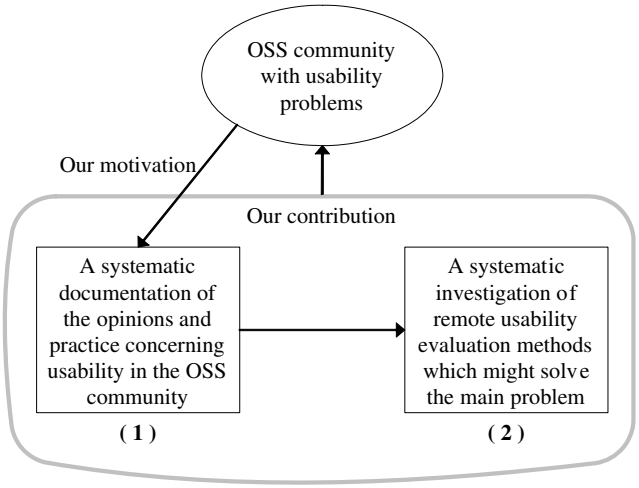

Figure 3.1: The relation between the two research papers. We were motivated by the problems in the OSS community and during the research conducted we suggest methods to improve the usability of OSS - our contribution. The numbers in the figure refer to the two papers below.

1. Andreasen, M. S., Nielsen, H. V., Schrøder, S. O. (2006).
   Usability in Open Source Software development: Opinions and practice.
   Department of Computer Science, Aalborg University, 2006

2. Andreasen, M. S., Nielsen, H. V., Schrøder, S. O. (2006).
   A Comparison of Remote Usability Evaluation Methods.
   Department of Computer Science, Aalborg University, 2006

3.1 Research Paper 1
Usability in Open Source Software Development: Opinions and Practice

The first paper describes a series of surveys concerning the subject of usability within OSS.

- Questionnaire survey
  - A questionnaire survey was performed in order to get an overview of the opinions and current practice of usability evaluation among OSS developers.
- Interviews with OSS developers
  - Interviews with three of the respondents of the questionnaire were performed to get a clear picture of the methods and thoughts connected to usability and evaluation in OSS projects. The three developers were selected because they were either project managers or usability evaluators in their current OSS project.
- Interviews with OSS evaluators
  - In order to get several perspectives on usability within the OSS community, we interviewed five usability professionals at the Berlin based usability company Relevantive. This company has extensive experience working with OSS projects and initiated the openusability.org project. We wanted to investigate what their experiences was working with OSS and usability.

We used interview guides based on the data from the questionnaires during the follow-up interviews. The themes in the interview guides included: Motivation for contributing to OSS, usability methods used in the project, usability as a part of the development process, usability experts in the development team, and decision making in the project. At Relevantive we performed a focus group interview followed by personal interviews. In the focus group interview we used an interview guide which included the following themes: Test procedures of OSS, usability evaluation, communication with OSS developers, remote usability evaluation, and OSS and usability.

We found that people within the OSS community were interested in usability and regarded it's importance as high. However, in practice the current usability effort among the developers was limited. The majority stated that they used formal or informal UI guidelines or inspiration from similar programs in the development of the UI. In the follow-up interviews we learned that the democratic and flat organizational structure of OSS development, which is often considered a strength, makes it a challenge to use conventional usability approaches in the development process. For instance there was reluctance to include usability experts directly in the development process because developers feared that they would overrule the democratic decision making process. In order to overcome the reluctance among developers it was necessary for usability professionals to build up trust. This was achieved after a history of sensible, realistic and constructive usability related suggestions and by meeting OSS developers in person at various conferences and gatherings. Furthermore, Relevantive stated that OSS developers who design user interfaces should be aware of basic usability principles. Currently there are limited resources available among usability experts involved with OSS and this makes it vital that developers do not rely entirely on external usability evaluations.

The paper lists a number of obstacles for usability work within the OSS community. In paper 2 we explore usability evaluation methods with the potential to solve these obstacles.

3.2 Research Paper 2
A Comparison of Remote Usability Evaluation Methods

The second paper looks at three approaches to remote usability evaluation. The methods were compared to a laboratory evaluation which was used as a benchmark. The motivation for researching remote evaluation methodologies was founded in the need for suitable evaluation methods for the OSS community. Furthermore, the area has not undergone substantial research and no methodical comparison of multiple remote evaluation methods has been made.

The methods evaluated were the following:

- Laboratory evaluation (LAB)
  - A laboratory evaluation based on the think aloud protocol described by Rubin [33] was performed and used as a benchmark.
- Remote synchronous evaluation (RS)
  - A remote think aloud evaluation using web-cams, audio connection, and a shared desktop to simulate a laboratory evaluation. The test participants and the moderator were separated in space.
- Asynchronous expert evaluation (AE)
  - An asynchronous evaluation where the test participants consisted of expert usability evaluators. The evaluation was conducted in natural settings and usability problems were reported through an online questionnaire. The test participants and the moderator were separated in space and time.
- Asynchronous user evaluation (AU)
  - This evaluation was conducted similar to the AE evaluation. The only difference was that the participants were mainstream users.

In order to minimize subjective bias we used a strict data analysis method (described in detail in Appendix C on page 59) as we were only three evaluators. By using the any-two agreement equation we showed that the resulting evaluator effect underlined the validity of our results.

The results showed that the RS evaluation performed as well as a conventional laboratory experiment. Statistical analysis showed that there was no significant difference between the number of identified problems. We also found that there was no significant difference in number of cosmetic, serious, and critical problems identified. The two methods were comparable and identified the same problems.
When comparing the AE evaluation with a laboratory evaluation we found that there was a significant difference in the total number of identified problems. There was also a significant difference in the number of critical problems identified but not in respect to serious and cosmetic problems.
In the comparison of the AU evaluation to the laboratory evaluation we identified a statistical

significance in the total number of identified problems. There was also statistical significance in respect to critical and cosmetic problems while there was no statistical significance in the number of identified serious problems.
We also wanted to compare the two asynchronous evaluations to see whether the experts were better at identifying problems than the non experts. We found that there was no statistical significance in the number of identified problems and severity.

The paper concludes that the RS evaluation is comparable with a LAB evaluation. The asynchronous method identifies fewer problems than the LAB evaluation. However, most of the problems identified in the asynchronous evaluations were critical.

CHAPTER 4

Comparison with Other's Remote Evaluation Experiences

This chapter provides a short, systematic overview on how our results compare to the existing knowledge on remote evaluation methods. It supplements paper 2 by providing an overview of claimed advantages and disadvantages of remote usability evaluation.

The comparative study used a laboratory evaluation as a benchmark and had the limitation that the RS evaluation was not performed in a natural environment. Therefore we do not have basis to evaluate claims about cost efficiency, technical difficulties and the difficulties of recruiting users.

Table 4.1 on page 17 presents an overview of the statements including those that we can not evaluate. In the following sections we present the advantages and disadvantages from the existing literature and conclude on whether our results support these statements.

Remote Synchronous Evaluation

Claimed Advantages of RS Evaluation

Identifies the same or more usability problems than laboratory evaluations
We found that the RS evaluation identified the same number of usability problems as the laboratory evaluation. The RS evaluation identified a total of 38 problems compared to the 35 found in the LAB evaluation and this difference was not statistical significant (p=0.6073). In addition, there was no significant difference in the distribution of cosmetic (p=1), serious (p=0.3498) or critical (p=1) problems identified.
Our results supported this statement.

Claimed Disadvantages of RS Evaluation

There is a longer task completion time
Our results did not show longer completion time of RS tests than in LAB tests. On average a LAB test lasted 22:10 minutes (SD=05:20) compared to the 22:30 (SD=03:31) of an RS test.
Our results did not support this statement.

The moderator may not be completely engaged in the test as there is no physical presence
In some instances the moderator was not fully engaged in the RS tests, compared to the LAB tests. For instance the moderator was not as good at encouraging the test user to think aloud as in the LAB evaluation. However this was not visible in the results and we do not have data to evaluate the claim.
Our results were inconclusive on this statement.

Distance between user and moderator can be awkward
We did not experience that the distance between the moderator and test user was awkward in the RS evaluation. Moreover, we did not notice misunderstandings because of the web-cam based communication and in the debriefing of the users, some expressed that they liked the fact that the moderator was not looking over their shoulder.
Our results were inconclusive on this statement.

The moderator looses control
The loss of moderator control in the RS evaluation, was not a problem in the majority of test sessions. In one of the tests we did experience that a user moved the web-cam windows and minimized others. This instance was solved by talking to the test user but visualized the lack of moderator control.
Our results were inconclusive on this statement.

Asynchronous Expert and Asynchronous User Evaluations

Claimed Advantages of AE and AU Evaluation

Users identify usability problems themselves resulting in lower workload for the evaluators
During the data analysis the evaluators used considerately less time generating problem lists for the individual AE and AU tests. We estimated the analysis time of each asynchronous test to be between 5 and 25 minutes. This was considerately less time than the evaluators used on analyzing the data from the LAB and RS evaluations. The analysis of this data took approximately 3 hours per test session (lasting on average 25 minutes per test session). The duration of the video analysis compares to the time spent in a study performed by Kjeldskov et al [21]. Some problems reported by users were not immediately understandable and needed to be interpreted by the evaluators, but overall the workload of the evaluators was lower than in the LAB evaluation.
Our results supported this statement.

The evaluation produces qualitative descriptions of problems
The quality of problem reports from users varied considerately. Most were very descriptive: "*I marked the Inbox and chose Functions –> Run filter on the chosen folder. That did not work - tried three times. Then I chose Functions –> Message filter –> marked the filter rule –> Run now on Inbox. I don't understand why the first action did not work. Run filter on the chosen folder sounds exactly as what I wanted to do.*". While others tended to be short and imprecise: "*It wasn't easy to do. I'm not that good at handling computers and I have never made a mail filter before.*". In general we found the quality of the descriptions to be good enough to be interpreted by the evaluators in order to generate problem lists.
Our results supported this statement.

Only minimal usability evaluation training of users is required
In the asynchronous evaluations we provided minimal training in the form of a classification matrix explaining how usability problems should be categorized as either cosmetic, serious, or critical. We also showed an example of a problem description. Though users did report problems without further training than this, most problems were not classified correctly compared to the evaluators characterization of the same problems. We therefore found that users needed more information about classification of usability problems than we provided.
Our results did not support this statement.

Very few critical issues missed by users
The AE evaluation identified 15 critical usability problems and the AU evaluation found 11. Compared to the 22 problems identified in the LAB evaluation, these numbers showed significant difference (p=0.0363 and p=0.0078).
Our results did not support this statement.

The evaluation is conducted in realistic test environments with external interruptions
We conducted the AE and AU evaluations in natural settings. The users performed the test at their own computers when it was convenient for them. Though we do not know whether this influenced test results we got feedback suggesting that this was a positive element: "*I think this method gives a better impression on how much people wants to work on a task before moving on to the next task.*". Overall we do not have enough information to assess whether testing in natural settings was an advantage in our study.
Our results were inconclusive on this statement.

Claimed Disadvantages of AE and AU Evaluation

Users need to be prompted to keep making comments while reporting critical incidents
Overall the test users reported a low number of usability problems compared to the time spent on the evaluation. In the AU evaluation users spent on average 45:29 minutes (SD=18:51) and reported 3.2 usability problems. The AE test users identified an average of 4.7 usability problems and spent on average 1:03:48 hours (SD=48:37) performing the test. The LAB evaluation identified on average 15.3 usability problems per test session with an average time consumption of 22:10 minutes (SD=05:20). It is important to remember that problems in the RS and LAB

evaluations were identified by the evaluators while they were identified by the test users in the AE and AU evaluations. These numbers indicate that the test participants with more prompting would have reported more problems, but we have no evidence of this.
Our results were inconclusive on this statement.

Methods	Advantages	Disadvantages
RS evaluation	+ Identifies the same or more usability problems than conventional laboratory evaluations [19] [7] [18] [38]	÷ There is a longer task completion time [38] ? The moderator may not be completely engaged in the test as there is no physical presence [1] ? Distance between user and moderator can be awkward [34] ? The moderator looses control [3] [4] [5]
RS evaluation (Not investigated in our study)	Cost effective in regard to travel expenses and accommodation [34] [16] [3] [4] [5] [19] [18] [10] A larger and more diverse pool of users is available [16] [3] [4] [5] [19] [18] It is easier to recruit users [19] [7] The evaluation is conducted in realistic environments with external interruptions [34] [16] [3] [4] [5] [38] [10]	Lack of facial expressions and body language when not using web-cam [34] [16] [3] [4] [5] [10] User must setup the system and there are technical demands in regard to bandwidth and server [34] [16] [3] [4] [5] Difficult to build trust between moderator and participant [10] The evaluation is conducted in realistic environments with external interruptions [16] [3] [4] [5] [7] The setup time is longer and there are difficulties in fixing system crashes [7] [10]
AE and AU evaluations	+ Users identify usability problems themselves resulting in lower workload for the evaluators [17] [8] + The evaluation produces qualitative descriptions of problems [18] ÷ Only minimal usability evaluation training of users is required [17] [8] ÷ Very few critical issues missed by users [17] [8] ? The evaluation is conducted in realistic environments with external interruptions [17] [8] [18]	? Users need to be prompted to keep making comments while reporting critical incidents [18]
AE and AU evaluations (Not investigated in our study)	Cost effective in regard to travel expenses and accommodation [17] [8]	Too long time before users report critical incidents, resulting in missing contextual information [17] [8]

Table 4.1: Advantages and disadvantages of RS, AE and AU evaluation methods identified in the literature. The symbols before each statement denote how our study relates to it: '+' = Our results agree on this statement, '÷' = Our results do not agree on the statement, and '?' = Our results are inconclusive on this statement. Furthermore, there are findings in the literature which our study has not investigated. The numbers refer to the literature in the bibliography.

CHAPTER 5

Research Methodology

This chapter describes and explains the research methods used to answer the two research questions. We have based this section primarily on the research conducted by Wynekoop and Conger [42]. They present an overview of ten different research methodologies of which we have used three.

The research methods are divided according to whether the research is performed in a natural or artificial setting (Table 5.1).

Research methodologies				
RQ	Purpose	Object	Method	Setting
#1	Describe and understand	OSS contributors	Survey research	Environment independent
#2	Comparison of methods	Remote usability evaluation methods	Laboratory experiment (LAB, RS)	Artificial
			Field experiment (AE, AU)	Natural

Table 5.1: Research methods used to answer the two research questions. Object refer to the object on which the method was used. The method is the chosen research methodology. The purpose describes the overall purpose of the study, while setting describes the setting the research was performed in.

5.1 Research Question 1

The research performed in paper 1 covered information on the opinions and practice towards usability among contributors to OSS (Section 1.1.1). We performed a questionnaire survey and two interview surveys.

According to Wynekoop and Conger, questionnaire and interview surveys belong to the category

of field surveys. Conducting a field study is preferable when the object of the study is to collect descriptive data and when developing or testing hypotheses. Wynekoop and Conger presents a number of advantages and disadvantages related to this research:

- Advantage
  - It is possible to collect large amounts of data which makes it possible to generalize on a subject while also reducing bias
- Disadvantages
  - It presents a static view of the domain
  - Participating is voluntary

The main advantage of conducting this type of research is the possibility to generalize on the data. In the questionnaire survey we received 24 answers out of a minimum of 293 possible answers which reduces the generalizability of our results. In an effort to minimize this negative side effect we did not perform statistical analysis on the data. The information was primarily used as inspiration when creating the interview guides used during the three developer interviews.

A disadvantage is connected to the fact that this type of survey often presents a static view of the domain. A possible solution to this is to perform a longitudinal study. Instead of doing this we found another study investigating the OSS community, and several of it's findings corresponded to ours. This study was performed by the Boston Consulting Group in 2002 and it revealed demographics, interests, and motivations of the OSS contributors [23].

We specifically pointed out that the questionnaire was directed at usability considerations and OSS and it is plausible that this have affected the bias of who participated in the questionnaire survey. When we conducted the interviews we selected the respondents in a way so that both usability testers and project managers were interviewed. The main point of this was to receive information on about usability considerations and project management but it was also done in an effort to minimize the bias of only interviewing contributors with usability responsibility.

5.2 Research Question 2

Following the findings in the first paper we made a thorough comparison of three remote usability evaluations to a laboratory evaluation (Section 1.1.2). When conducting the experiments we used two different research methodologies. The LAB and RS evaluations were performed as laboratory experiments while the AE and AU evaluations were field experiments.

Laboratory Experiment

According to Wynekoop and Conger there are the following advantages and disadvantages related to a laboratory experiment:

- Advantages
  - Experiments can be replicated because of the large experimenter control

  - It provides a precise measurement of the phenomena of interest
- Disadvantages
  - The method assumes that real-world interference is not important
  - Generality is limited to the sample population

The goal of our research was to evaluate three approaches to remote usability evaluation methods and not an application. In this respect, replicability is a priority so other researchers can use the experience collected through the study. The increased level of experimenter control made it possible to make sure that the setup worked properly and that the individual test sessions were recorded in the correct way.

In the evaluations performed in the laboratory we were able to measure the exact time used to perform the evaluation which provided data for statistical analysis. Moreover, we had the possibility to observe the participants while they conducted the test.

We chose to conduct the RS evaluation in a controlled environment as opposed to the AE and AU evaluations conducted in natural settings. We performed it in a laboratory since we wished to be able to control the setup and make sure that there were no technical difficulties relating to the setup or bandwidth.

We used a sample consisting of students from Aalborg University. Moreover, we selected participants who had not used the tested application but at the same time were familiar with the central concept of e-mail. In order to minimize the problem of generality, a replication of the laboratory evaluation using a different sample could have been performed.

Field Experiment

According to Wynekoop and Conger there are the following advantages and disadvantages related to a field experiment:

- Advantages
  - It is carried out in natural settings
  - There is increased control over variables compared to a field study
- Disadvantages
  - Increased experimenter control over variables results in a decrease of the naturalness
  - There is less experimenter control over possible contaminating effects

When we analyzed the results of the asynchronous evaluations compared to the results from the experiments in the laboratory, we saw that the asynchronous participants solved insignificantly more tasks while using longer time. This could be a result of the differences of the settings. For instance, the participants was not affected by a moderator. One expert participant of an asynchronous evaluation even stated that this method was better at showing how long a user wanted to work with a task before giving up.

We chose to minimize our control over the variables. Even though this control is presented as an advantage of the field experiment it is also presented as a possible disadvantage. In this

respect we chose to control only the most crucial variables: How problems were reported using the questionnaire and which program to test.

We controlled some variables during the evaluation while the participants controlled others. For instance the participants were free to conduct the evaluation when and how they wanted. Hereby we increased the naturalness of the AE and AU evaluations.

As the evaluations were performed in natural settings we lost control over possible contaminating effects. We chose to perform the AE and AU evaluations as field experiments since the technical problems, which discouraged us from conducting the RS evaluation as a field experiment, were not present here.

CHAPTER 6

Conclusion

The focus of this master thesis is usability in OSS projects. In this chapter we present the conclusions for the research questions of the two papers. Subsequently we relate these answers to the overall research question (Section 1.1) and present the limitations and possible future work.

Research question 1: How is usability currently considered by both OSS developers and usability experts participating in OSS projects, what is the current practice of usability evaluation within OSS development, and what are the obstacles for change?

The first research paper showed that even though OSS contributors were interested in usability and regarded it's importance as high in their projects, in practice the usability effort was limited. Most usability considerations were based on common sense, design guidelines and conventions from other programs. Expert inspections and conventional usability evaluation were also used to some extent but the development model, and the lack of knowledge among OSS contributors, made this problematic. OSS projects are characterized by a flat organizational structure based on democratic principles, and contributors expressed concern that direct involvement of usability experts would disrupt the balance of power in internal discussions. We found that the most promising current approach to usability evaluation in OSS was conducted by groups of external usability professionals, as the results of their evaluations were appreciated and to some extent used by the developers. However, there are only a limited number of contributors with the necessary knowledge about usability, few have access to usability laboratories and the distributed nature of OSS makes it a challenge to co-operate with developers. Usability professionals involved with OSS also mentioned the obstacle of recruiting volunteer test users to participate in usability evaluations. Therefore we found that conventional usability evaluation methods did not support the OSS development paradigm and we wanted to investigate whether alternatives existed that solved the following obstacles:

- The lack of access to usability laboratories
- There is a limited number of usability experts available
- The lack of financial resources
- The geographically distributed nature of OSS
- The problem of recruitment of test users
- The short development cycles of OSS development
- The preference of externally performed usability evaluations by contributors

We chose to focus the second research question on remote usability evaluation in order to explore whether this type of evaluation would overcome some of the key obstacles.

Research question 2: How do three different approaches to remote usability evaluation compare to a laboratory evaluation in regard to the identification of usability problems?

The second research paper revealed that the results of the RS evaluation were comparable to those of the LAB evaluation. The two evaluation methods identified the same number of problems and almost the same distribution of cosmetic, serious and critical problems. Some test users even felt more comfortable in an evaluation setup, where the moderator was not physically present. Hence, an RS evaluation can be as effective as a LAB evaluation.

In the AE and AU evaluation methods, we did not find any difference between expert users and ordinary users in their ability to identify usability problems. In both evaluations users identified fewer problems than the LAB evaluation, and their classification of problem severity did not match the classification of the evaluators. Almost all problems were classified cosmetic by the users but when reviewed by the evaluators, most of these problems turned out to be critical. Thus, if a user in a remote asynchronous evaluation reports a problem, it is fair to assume that it is indeed a critical problem. This type of evaluation can therefore be performed quantitatively with large samples of a system's user base in order to identify the most severe usability problems of the system. It can also be used as a supplement to other types of evaluation or as low cost alternative for those without the resources to perform a conventional usability evaluation.

Overall research question: How can the usability work in OSS development be improved?

We found that there is an interest in an increased level of usability among those involved with OSS development. However, lack of knowledge about usability among many developers is a fundamental problem in the process of designing user interfaces. It is also a challenge for the usability effort within OSS that development is distributed geographically since face-to-face communication with development teams usually is not possible.

Both developers and usability professionals prefer that usability evaluations are performed by people with in-depth usability knowledge. Therefore, it is vital that the OSS community attracts this type of people and makes them feel that their contributions are appreciated. In practice there are not a lot of usability experts involved with OSS and furthermore they do not all have access to usability laboratories and volunteer test users. We found that the RS evaluation method can simulate a usability laboratory and generate comparable results. Thus, we consider this method a possibility for usability experts who do not have the resources to perform a laboratory test or those who need volunteer test users in certain narrow user segments that are not available locally.

The asynchronous evaluation method is a usable method within the OSS community. Since users report and classify usability problems themselves this type of evaluation can be performed by the developers without assistance from the scarce number of external usability experts. Given that the problem descriptions contain a satisfactory level of detail, the developers can use these reports directly in the development process.
In this thesis we have investigated how usability work in OSS can be improved and we have suggested approaches to remote usability evaluation as a possibility for solving the key obstacles. There are other fundamental usability related challenges for the OSS community than the evaluation methods but this is beyond the scope of this study.

The questionnaire survey received twenty-four responses from a potential user group with a minimum of 293 possible. Two of the developers in the follow-up interviews were associated with the same project. We performed the LAB and RS evaluations in the controlled environment of a usability laboratory, while the AE and AU evaluations were performed in natural settings.

We see three obvious subjects for further research. First of all, it would be interesting to further develop the asynchronous method. The method has potential to solve many obstacles as the identification of problems is placed at the user. The main problem concerning this method was the low number of identified problems. Providing the test participants with a better conceptual tool for identifying problems could solve this. Furthermore, it would be relevant to conduct the RS usability evaluation in a natural environment in order to reflect over the technical challenges of this method. Finally, it would be compelling to let experienced usability evaluators assess the RS evaluation method, in order to get their qualitative impression of the method.

Bibliography

[1] Ames, M.
final report on remote usability studies.
http://www.ocf.berkeley.edu/~morganya/research/dmp/report.html,
2005.

[2] Ballmer, S.
interview with chicago sun times.
Chicago Sun Times
http://www.suntimes.com/output/tech/cst-fin-micro01.html,
2001.

[3] Bartek, V., and Cheatham, D.
experience remote usability testing, part 1.
www-106.ibm.com/developerworks/library/wa-rmusts1/,
January 2003.

[4] Bartek, V., and Cheatham, D.
experience remote usability testing, part 2.
www-106.ibm.com/developerworks/web/library/wa-rmusts2.html,
February 2003.

[5] Bartek, V., and Cheatham, D.
experiences in remote usability evaluations.
http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/
50?OpenDocument&../Publish/1116/$File/paper1116.pdf,
2004.

[6] Benson, C., Muller-Prove, M., and Mzourek, J. Professional usability in open source projects: Gnome, openoffice.org, netbeans. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems* (New York, NY, USA, 2004), ACM Press, pp. 1083–1084.

[7] Brush, A. B., Ames, M., and Davis, J. A comparison of synchronous remote and local usability studies for an expert interface. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems* (New York, NY, USA, 2004), ACM Press, pp. 1179–1182.

[8] Castillo, J. C., Hartson, H. R., and Hix, D. Remote usability evaluation: can users report their own critical incidents? In *CHI '98: CHI 98 conference summary on Human factors in computing systems* (New York, NY, USA, 1998), ACM Press, pp. 253–254.

[9] CNN.
reclusive linux founder opens up.
http://edition.cnn.com/2006/BUSINESS/05/18/global.office.linustorvalds/,
May 2006.

[10] Dray, S., and Siegel, D. Remote possibilities?: international usability testing at a distance. *interactions 11*, 2 (2004), 10–17.

[11] Eklund, S., Feldman, M., Trombley, M., and Sinha, R.
improving the usability of open source software: Usability testing of staroffice calc.
http://www.sims.berkeley.edu/~sinha/opensource.html,
2001.

[12] Feller, J., and Fitzgerald, B. *Understanding open source software development.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[13] Fogel, K. *Producing Open Source Software - How to run a succesful Free Software Project.* O'Reilly Media, 1005 Gravenstein Highway North, Sebastopol, CA 95472, USA, 2006.

[14] Frishberg, N., Dirks, A. M., Benson, C., Nickell, S., and Smith, S. Getting to know you: open source development meets usability. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems* (New York, NY, USA, 2002), ACM Press, pp. 932–933.

[15] Gates, W. H.
open letter to hobbyists.
http://en.wikipedia.org/wiki/Open_Letter_to_Hobbyists,
1976.

[16] Gough, D., and Phillips, H. Remote online usability testing: Why, how, and when to use it.
http://www.boxesandarrows.com/view/remote_online_usability_testing_
why_how_and_when_to_use_it,
2003.

[17] Hartson, H. R., and Castillo, J. C. Remote evaluation for post-deployment usability improvement. In *AVI '98: Proceedings of the working conference on Advanced visual interfaces* (New York, NY, USA, 1998), ACM Press, pp. 22–29.

[18] Hartson, H. R., Castillo, J. C., Kelso, J., and Neale, W. C. Remote evaluation: the network as an extension of the usability laboratory. In *CHI '96: Proceedings of the*

SIGCHI conference on Human factors in computing systems (New York, NY, USA, 1996), ACM Press, pp. 228–235.

[19] Houck-Whitaker, J.
remote testing versus lab testing.
http://boltpeters.com/articles/versus.html,
2005.

[20] Kavanagh, P. *Open Source Software - Implementation and Management*. Elsevier Digital Press, 200 Wheeler Road, Burlington, MA 01803, USA, 2004.

[21] Kjeldskov, J., Skov, M. B., and Stage, J. Instant data analysis: conducting usability evaluations in a day. In *NordiCHI '04: Proceedings of the third Nordic conference on Human-computer interaction* (New York, NY, USA, 2004), ACM Press, pp. 233–240.

[22] Kjeldskov, J., Skov, M. B., and Stage, J. Does time heal : a longitudinal study of usability. In *OZCHI '05: Proceedings of the 19th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction* (Narrabundah, Australia, Australia, 2005), Computer-Human Interaction Special Interest Group (CHISIG) of Australia, pp. 1–10.

[23] Lakhani, K., and Wolf, B.
bcg hacker survey.
Boston Consulting Group
http://www.ostg.com/bcg/BCGHACKERSURVEY-0.73.pdf,
2002.

[24] McBride, D. Open letter on copyrights.
http://www.caldera.com/copyright/,
December 2003.

[25] McBride, D. Letter to the congress.
http://www.osaia.org/letters/sco_hill.pdf,
January 2004.

[26] Mockus, A., Fielding, R., and Herbsleb, J. Two case studies of open source software development: Apache and mozilla. *ACM Trans. Softw. Eng. Methodol. 11*, 3 (2002), 309–346.

[27] Mundie, C. The commercial software model.
http://www.microsoft.com/presspass/exec/craig/05-03sharedsource.mspx,
May 2001.
Speech at The New York University Stern School of Business.

[28] Nichols, D. M., and Twidale, M. B. Usability and open source software. Tech. Rep. 10/02, Department of Computer Science, University of Waikato, 2002. Working Paper Series ISSN 1170-487X.

[29] Opensource.org.
shared source: A dangerous virus.
http://www.opensource.org/advocacy/shared_source.php,
2002.

[30] Palmisano, S.
ibm's annual stockholders meeting.
IBM website
http://www.ibm.com/ibm/sjp/04-30-2002.html,
2002.

[31] Perens, B.
open source initiative - the open source definition.
http://www.opensource.org/docs/definition.php,
1997.

[32] Raymond, E. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary.* O'Reilly, feb 1999.

[33] Rubin, J.
Handbook of Usability Testing.
Katherine Schowalter, 1994.

[34] Safire, M.
remote moderated usability.
http://www.upassoc.org/usability_resources/conference/2004/im_safire.html,
2004.

[35] Stallman, R.
gnu general public license.
http://www.gnu.org/copyleft/gpl.html,
1991.

[36] Stallman, R.
why 'free software' is better than 'open source'.
http://www.gnu.org/philosophy/free-software-for-freedom.html,
2001.

[37] Stallman, R.
gnu operating system - free software foundation.
http://www.gnu.org,
2004.

[38] Thompson, K. E., Rozanski, E. P., and Haake, A. R. Here, there, anywhere: remote usability testing that works. In *CITC5 '04: Proceedings of the 5th conference on Information technology education* (New York, NY, USA, 2004), ACM Press, pp. 132–137.

[39] Trudelle, P. Shall we dance? - ten lessons learned from netscape's flirtation with open source ui development. In *CHI 2002* (2002), Presented at the Open Source Meets Usability Workshop, Conference on Human Factors in Computer Systems.

[40] Valloppillil, V., and Cohen, J.
microsoft: The halloween documents.
http://www.catb.org/~esr/halloween/,
1998.
Microsoft has acknowledged the authenticity of these leaked internal memos.

[41] Vixie, P. Software engineering. In *Open Sources: Voices from the Open Source Revolution* (New York, NY, USA, 1999), O'Reilly.

[42] Wynekoop, J. L., and Conger, S. A.
a review of computer aided software engineering research methods. In *Proceedings of the IFIP TC8 WG 8.2 Working Conference on The Information Systems Research Arena of The 90's* (1990).

APPENDIX A

Usability in Open Source Software Development: Opinions and Practice

Usability in Open Source Software Development: Opinions and Practice

Morten Sieker Andreasen
Department of Computer Science
Aalborg University, Denmark
sieker@cs.aau.dk

Henrik Villemann Nielsen
Department of Computer Science
Aalborg University, Denmark
villeman@cs.aau.dk

Simon Ormholt Schrøder
Department of Computer Science
Aalborg University, Denmark
ormholt@cs.aau.dk

ABSTRACT

Open Source Software developers have been able to produce programs that in functionality are able to compete with proprietary software. However, the programs are often designed for power-users and with little emphasis on usability. A questionnaire survey and a series of interviews with OSS contributors were performed in order to explore opinions on usability within OSS and current practice of the usability effort within OSS development. Contributors involved with OSS with both technical and usability backgrounds were interviewed. There are several obstacles for usability work within OSS such as geographically distributed development, lack of resources and evaluation methods that fit into the OSS paradigm.

Keywords

Open source software development, usability, usability evaluation, empirical study

1. INTRODUCTION

Open Source Software (OSS) is a concept that covers software which is released under a license that allows the end user to freely use, distribute and modify the source code of the program. OSS is often described as 'free' software, which refers to the liberty and not to the price of the software [13, 27]. Famous OSS projects include Linux, Eclipse, Apache, Tomcat and the Mozilla suite of programs. Although many companies like IBM, SUN and Novell are involved with OSS, the majority of OSS projects are developed by individuals or small development teams with little formal organization [16].

It has been claimed that use of the OSS development model results in increased security and quality, since the code is exposed to extreme scrutiny with problems being identified and solved swiftly [20]. In addition, the release cycles of OSS development projects are often short and allow rapid software development. Though the OSS community has produced software products able to compete with commercial alternatives [26, 9], OSS development also faces a number of fundamental challenges. The stereotype of OSS developers is that they represent the "*cowboys of the software world with few formal procedures, actively hostile to any authority other than the hacker ethic*" [9]. The premises of OSS are often vastly different from traditional software development environments; developers often work in their spare time and most communication needs to be by electronic means since developers often are distributed around the world. Eric Raymond, who is a prominent voice in the open source movement and co-founder of the Open Source Initiative, compared the development process of OSS to "*a great babbling bazaar of different agendas and approaches*" [20]. We chose to focus our research towards projects developed in small groups by volunteers. Corporations involved with OSS often have trained usability experts and laboratories at their disposal, and arguably their usability evaluation procedures do not differ to those of conventional software development.

In recent years there has been an increasing attention towards the technological achievements of OSS [14]. At the same time OSS has been criticized for it's lack of usability for non-technical users compared to commercial software [3, 6, 9, 17, 21]. There has been little research about the reasons that OSS generally does not have the same degree of usability as commercial software, and if the state of affairs regarding usability is to change, the core problems need to be identified. In this article we answer how usability is currently considered by both OSS developers and usability experts participating in OSS projects, what is the current practice of usability evaluation within OSS development, and what the obstacles for change are.

Addressing the key question, this paper describes a number of empirical studies performed to outline the problems and opportunities involved with the usability effort within the OSS community. In section 2 we describe how this paper relates to existing research within the field of OSS and usability. In section 3 we describe the methods we used to conduct a questionnaire survey and a series of interviews with OSS developers and usability professionals that had extensive experience working within the OSS community. In section 4 we present the main results of the empirical studies, and in section 5 we discuss three overall themes related to OSS based on the results of our study. Finally, we summarize the main results and answer the key question in the conclusion in section 6.

2. RELATED WORK

OSS has been the subject of several studies with different focus areas but little research has been conducted regarding usability evaluation or usability's role in the OSS development process. Feller and Fitzgerald described the OSS development paradigm and stated that the OSS development model could potentially solve some of the software industry's problems regarding reliability, pace of development and

cost, but their study did not suggest how [7]. Johnson-Eilola constructed a basic overview of issues of OSS models for development and distribution of computer documentation. His arguments concerned two different methods of implementing open source models for computer documentation but it was not very methodic and seemed biased [14]. Gasser and Ripoche conducted quantitative studies of the problem management and bug reporting methods within OSS. They extracted large amounts of data from the Bugzilla bug reporting tool and analyzed it in order to describe the process that a problem went through from the initial report to it's resolution [11]. Sandusky et al followed this study up with an investigation of the relations between bug reports, so called bug reporting networks, and an analysis of a single social negotiation [25, 24]. However, these studies were primarily focused on the resolution of technical problems and they did not cover the process of resolving usability problems. The Boston Consulting Group conducted a large, quantitative study of the background and interests of OSS contributors, which revealed among other things demographics, interests and motivations of a large sample of respondents [16].

Some work has been done to explore the field of usability in OSS. For instance Nichols and Twidale explored usability discussions in OSS development by examining the communication between developers who used a bug reporting utility for OSS. They described, how existing human-computer interaction techniques could be used to leverage distributed networked communities, of developers and users, to address issues of usability [30]. This study was based entirely on quantitative measures and ignored other forms of communication than the bug reporting utility. In another study they listed some of the general usability issues and challenges that OSS was facing [17]. They did not perform a systematic study of the OSS research area and they did not conduct any experiments to find solutions to the usability problems of OSS.

Trudelle described ten lessons that Netscape learned from it's flirtation with OSS user interface development. The presented experiences origins from just one company involved with OSS but on the other hand Trudelle describes down-to-earth considerations for companies. When Netscape released it's web-browser as OSS in the Mozilla project, they discovered that the design stage could not be skipped. Furthermore, Netscape found out that a division of authority and 'ownership' of various parts of the user interface was important in order to settle discussions and make decisions, yet open discussion and exposure of all possible work was advantageous. Conventional practice like identification of target users was also important since OSS contributors often worked with their own use in mind [29]. This was supported by Pemberton who stated that it was an obstacle for OSS's mainstream acceptance that developers with programming backgrounds did not understand the problems of ordinary users [19]. Eric Raymond stated that the OSS community needed *"a big player with a lot of money, which is doing systematic user interface end user testing. We're not very good at that yet, we need to find a way to be good at it"* [22]. According to Raymond, the problem concerned organizing usability evaluations and the costs of conducting them. As a key figure in the OSS community, Raymond must be considered biased. Nichols and Twidale agreed, and pointed to the lack of usability expertise and resources in the OSS community as the main problem [17].

Our motivation for writing this article was based on an interest in the OSS development paradigm. We were curious about investigating, whether the OSS development model was suitable for the design of intuitive user interfaces for anyone outside the 'hacker' community.

3. METHOD

In order to answer the key question we chose to conduct a series of empirical studies. We wanted to know more about the current practice of usability evaluation methods in OSS projects, and furthermore clarify the thoughts and experiences of usability evaluators working on OSS projects. This resulted in a total of three empirical surveys; an online questionnaire survey, interviews with three OSS developers and interviews with five people working at Relevantive, a company performing pro bono usability evaluation for OSS projects.

3.1 Questionnaire

Setting: The questionnaire survey was conducted from mid September to mid October 2005. The survey consisted of three parts, corresponding to areas we wanted to explore: 'About your current project', 'Communication' and 'Usability'. This was done to keep a relation between the questions, give the respondents an overview of the survey, and to make sure that the respondents understood what the intentions of the questions were [18]. The questions were mixed between quantitative scales where it was possible to choose from a set of predefined options, and qualitative questions where the respondents could answer freely.

Participants: To exploit the advantages of a quantitative survey we wanted to get a high number of respondents, and therefore we contacted fourteen different OSS projects, where the number of contributors varied from one person to more than fifty persons per project. We used two main criteria for selecting OSS projects to contact; they should not be developed by professionals and the product should be targeted at mainstream users. The reason for this selection was that we wanted to get feedback from developers who contributed to OSS software that was directed towards ordinary users and not technical users. This resulted in a total of twenty-four respondents located in fourteen different countries (Table 1). Via mailings lists and websites related to the projects, we found that the minimum number of contributors that received the invitation to participate in the questionnaire was 293.

Materials / Data collection: For the online survey we used the PHP/MySQL based survey tool UCCASS. This made it simple to extract the data in a variety of data formats, once the survey had been completed.

Data Analysis: The data analysis was conducted in two parts. First we extracted all the data from the online questionnaire, mapped the quantitative data in diagrams and graphs and organized all the qualitative data in sections reflecting the three research areas of the questionnaire (Figure 1). We performed the actual analysis through use of condensation of meaning. This method covers the idea that the

Project	Contacted	n	Description	c	Contributor countries
Mplayer	yes	1	A multimedia player for Linux	16	Germany
The GIMP	yes	5	An image editing program	6	USA, Germany, Australia, France
Kopete	yes	5	An instant messaging client	6	Brazil, Chile, Germany, UK, Turkey
GNU cash	yes	1	A budget program for Linux	1	UK
Gnome	yes	1	Desktop environment	50	Denmark
Abiword	yes	1	An open source word processor	11	UK
mmsv2	yes	2	A system to play multimedia files on a TV	1	Denmark, Sweden
Gnumeric	yes	2	A spreadsheet application for Gnome	1	Denmark, Australia
Xine	yes	1	A multimedia player	11	Germany
Inkscape	yes	0	A vector drawing program	16	
Konqueror	yes	0	KDE's file manager and browser	33	
Kile	yes	0	A LaTeXsource editor	29	
K3B	yes	0	A CD-ROM / DVD burning application	4	
Amarok	yes	0	A multimedia jukebox	5	
Point of sale	no	1	Proprietary retail inventory software	1	Belgium
Tuxpaint	no	1	A children's painting program	50	USA
Tapper	no	1	A software installation manager for Linux	1	Norway
Scribus	no	1	A Desktop Publishing application	50	Canada
Kshower	no	1	A visualization tool for data	1	Greece
Total		**24**		**293**	

Table 1: Overview of contacted projects and those who chose to participate in the survey - n indicates the number of respondent from the particular OSS project - c depicts the minimum number of contributors in the project.

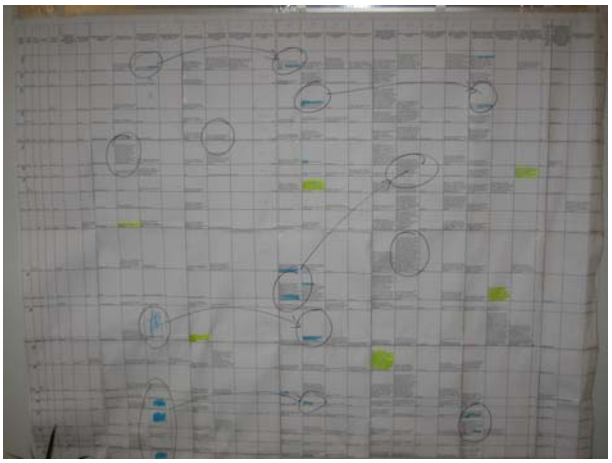

Figure 1: A mapping of the questionnaire results

researcher concentrates on the essential parts of an answer [15] We used the data to determine tendencies and focus areas for the follow-up interviews with the OSS developers.

3.2 Developer Interviews

Setting: To get a clear picture of the methods and thoughts connected to usability evaluation in OSS projects, we chose to perform interviews with three of the respondents from the questionnaire. In the questionnaire we asked the respondents if they were willing to help us further by participating in a personal interview, and eighteen out of the twenty-four respondents answered that we could contact them for more information about their project.

Participants: Since the main purpose of the interviews was to acquire knowledge about usability within open source software we looked for respondents who had the role of usability tester or project manager. The role of project manager was relevant since we wanted to attain knowledge on how OSS projects were administered in regards to usability. In the questionnaire respondents selected the various roles that they had in their OSS project. We used this to select respondents, who had indicated that they were - among other roles - project managers or usability testers, for follow-up interviews. This screening process resulted in the selection of three OSS contributors for the follow-up interviews; two project managers and one usability tester. One of the project managers and the usability tester were both involved with the Kopete instant messaging application and the second project manager was the main developer on mmsv2, a system to organize and play back multimedia files on a TV.

Materials: The interviews were conducted as semi-structured interviews, based on an interview guide to make the results comparable. As proposed by Steiner Kvale [15], the interview guide was based on a set of thematic questions. The thematic questions were made to ensure that the interviews investigated the same areas of interest, and thereby made the results comparable. The research question lead us to investigate the following themes:

- Respondent's motivation for contributing to OSS
- Usability considerations used in the project
- Frequency and place of usability evaluations
- Usability as a part of the development process
- Usability experts in the development team
- Willingness to alter program code because of usability problems discovered in tests
- Decision making in the project especially in regards to usability

Data Collection: As the respondents were located in three different countries we contacted them separately and suggested interview methods according to what was possible and convenient for them. This resulted in one in person interview and two interviews via instant messaging software. In the in person interview we used an audio recorder for our data collection. We later transcribed the recording, to ease our analysis. The process of transcribing took 10 man hours and resulted in 9 pages of text. The interviews conducted through instant messaging were transcribed through a built-in feature, that saved the conversation in an XML file.

Data Analysis: In the data analysis of the interview we also used condensation of meaning. First we identified a number of topics or tendencies we found important in the transcriptions. Following this we analyzed the statements in more detail to extract the overall opinion of the respondent.

3.3 Evaluator Interviews

Setting / Participants: To get a another perspective on how usability tests were conducted in an open source context, we contacted the company Relevantive, located in Berlin. At the moment Relevantive is one of the leading forces within usability in open source projects and they develop and administer the website openusability.org, where they provide communication between open source projects and usability evaluators.

Materials: We performed the interviews over a six hour span, where we first conducted a two hour focus group interview with the five employees. Hereafter, we had personal interviews with two of the employees and finally we had the opportunity to observe them while they conducted a usability test of an OSS product. The focus group interview was conducted as a semi structured interview, based on an interview guide with seven themes we wanted to discuss:

- Background information about Relevantive
- Test procedures of OSS
- Usability evaluation
- Communication with OSS developers
- K Desktop Environment (KDE) guidelines
- Remote usability evaluation
- OSS and usability in general

For the individual interviews we did not use a pre-constructed interview guide, but instead we used the data collected from the focus group interview to find new themes we wanted to explore further.

Data Collection: We used a combination of audio recording and note taking to collect data. The focus group interview was recorded on a laptop, and later transcribed. The process of transcribing took 16 man hours and produced 15 pages of text. The individual interviews were not recorded, but all three moderators took notes. Immediately after the individual interviews the three moderators compared notes and gathered these in a single text document. This process took 2 hours and resulted in 6 pages of text.

Data Analysis: For the data analysis we used the same procedure as in the developer interviews.

4. RESULTS

Through the empirical studies we attained knowledge on the perception of usability in OSS by both developers and usability professionals. Furthermore, we got an insight in the current practice of usability evaluation in OSS projects. The results covered the following themes:

- The OSS contributors
- Opinions about usability
- The OSS development process
- Usability evaluation methods in practice

4.1 The OSS Contributors

Contributors to OSS are highly ideological. Initially we wanted to investigate who the OSS contributors were and what their motivation for participating in OSS development was. In the questionnaire the respondents primarily consisted of males between 19 and 40. This is comparable to the Boston Consulting Group survey that found the average age of OSS contributors to be 30 years [16]. In the questionnaire and in the interviews we found that the main motivation for contributing to OSS for both developers and usability experts was ideology. There was a general feeling that contributing to OSS was the right thing to do. In the questionnaire answers 88 per cent of the developers chose 'To strengthen free software' as their motivation (Figure 2) and the interviews with developers and usability professionals supported this. Another finding was that only one person was motivated because he was getting paid by an employer. This underlined the ideological foundation we found OSS developers to base their contribution on. In addition, 54 per cent of the questionnaire respondents chose 'Community reputation' as a motivation. We consider the people who prioritized strengthening of free software or community reputation to be the ones who potentially would have an interest in increasing the market share of of OSS by making it more accessible to mainstream users.

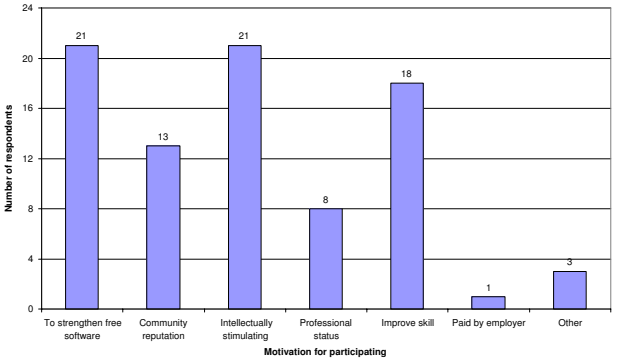

Figure 2: The motivation for developers to contribute to OSS. The 24 respondents could choose more than one motivating factor.

Contributors to OSS want challenges. The results of the questionnaire showed that 75 per cent of developers contributed to OSS in order to improve their skills and 88 per

cent wanted to be intellectually stimulated. The technical challenge as a motivating factor was also evident in the interviews with OSS contributors. A project manager explained that the idea for his OSS project came about "*because at the time when I started it, there weren't any multimedia systems actually that was either open source or closed software*". Another project manager explained that he initially needed a program to chat with his girlfriend and therefore wanted to develop a system that allowed him to do so. One third of the questionnaire respondents furthermore stated that they were motivated to contribute to OSS because of 'Professional status'. We interpreted this to be similar to improvement of skill in the way that OSS contributors feel that they develop themselves and gain knowledge by their involvement in OSS. The interviewed usability professionals explained that they got valuable experience from their involvement with OSS. The immediate effect that a usability evaluation resulted in was very satisfying and a great motivation for further involvement in OSS. For instance the results of Relevantive's usability evaluation of the German edition of Wikipedia, an open source on-line encyclopedia, were also implemented in the international edition of Wikipedia. This differed greatly from their co-operation with commercial software companies, where consultant reports often resulted in only minor changes to the software - or none at all.

In short, OSS contributors are mainly motivated by an ideological belief in free software and to be intellectually stimulated. Developers as well as people with usability expertise also feel that they receive valuable skills and experience from their involvement with OSS and they enjoy the challenges involved with OSS development.

4.2 Opinions about Usability

We wanted to investigate whether OSS contributors were positive towards usability and how much they knew about it. Hence, we explored this subject in the interviews and in the questionnaire survey.

In principle usability is important. The majority, a total of 83 per cent, of questionnaire respondents regarded the importance of usability as either 'high', 'very high' or 'extremely high'. Only 13 per cent considered it 'moderate', 4 per cent stated 'slight' and nobody thought it had no importance (Figure 3). Two of the OSS developers that we interviewed stated that usability had extremely high importance in the questionnaire. One of them explained that some developers see usability as a trivial task which is not interesting nor intellectually stimulating; "*...hackers code for fun, and sure it is more fun to add support for some protocol feature than fixing a dialog for grandma*". The third developer we interviewed answered that usability had only slight importance in the questionnaire. In the interview we investigated this and the main reason was that he developed the system as a hobby and that a large user base was not his goal.

The people at Relevantive considered the importance of usability extremely high. Their experience was that especially the young developers had usability as a high priority while some of the older developers were reluctant. We did not have enough information about the respondents in the questionnaire to support this notion though.

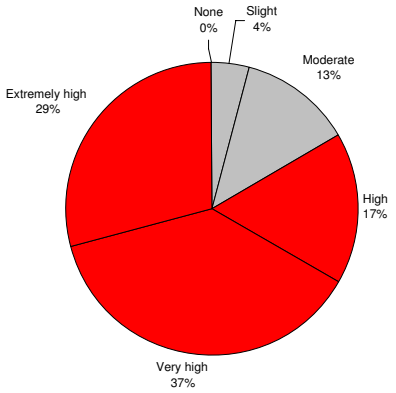

Figure 3: The priority of usability for the OSS developers in the questionnaire.

Although the questionnaire revealed a high priority of usability among OSS contributors it should be considered, whether these good intentions are in fact used in practice. The interviews showed that some developers have a pragmatic view on usability and find it more interesting to develop new features than correcting usability problems in the user interface.

Focus area	**Key words**	**n**
Effectiveness	Accuracy, completeness, productivity	7
Efficiency	Learning time, intuitiveness, resources spent	16
Satisfaction	Attitude to system, entertainment value	1
Technical property	Functionality of system	3
No category	Did not provide definition	1
Total		**28**

Table 2: The focus areas for the 24 definitions of usability. The value n = 28 is a result of 4 definitions that had multiple focus areas.

There is confusion about the definition of usability. Though OSS contributors wished to develop software which was easy to use, the definitions of the term 'usability' in the questionnaire varied considerately. We divided the definitions into three focus areas corresponding to the ISO9241 definition: Effectiveness, Efficiency and Satisfaction [10]. The definitions that did not relate to any of these focus areas were categorized as either 'Technical property' or 'No category' (Table 2). Efficiency was the center of focus for 16 of the definitions, for instance "*Usability is the science that's concerned with how quickly/easily a user is able to perform useful tasks with a given system*". Some of these definitions were focused on intuitiveness and put emphasis on the affordance of the user interface and user centered design: "*A user should be able to use the basics of the program without any help of documentation. This is basically done by building a GUI which maps to the users mind space. like putting a dustbin where you're gonna delete files*". Another respondent defined usability very brief but still emphasized the importance of intuitiveness "*If your grandma can use it*". Seven definitions were instead focused on effectiveness.

For instance one respondent stated that usability was: "*Allowing a user to perform tasks with as little diversion as possible*". Three of the OSS contributors who participated in the questionnaire defined usability based on some technical attribute of the program. One of them defined usability "*Within development: 1. Tools that do not get in the way of development. 2. Security through the use of GnuPG. After development: 1. Documentation. 2. Accessible bug reports. 3. Responding to users. 4. Internationalization. 5. Localization*" another simply defined usability as "*A working thing, that works when requested to work*". These definitions revealed an alternative, developer centered understanding of the term usability.

Overall a high number of the OSS contributors who participated in the questionnaire survey showed an understanding of at least one dimension of the term usability. Most emphasized efficiency as the most important aspect of usability and only few definitions covered more than one of the focus areas efficiency, effectivity, and satisfaction. None of the definitions of usability included all three focus areas and some definitions were simplified and arguably not easy to use in practice. Finally, three definitions were focused on aspects outside the scope of the ISO definition of usability. This confusion must be taken into consideration in relation to the high priority of usability among OSS contributors that we identified.

Usability experts are only advisors. Though most OSS contributors wanted a higher degree of usability in their software, they were reluctant to include usability experts directly in the development process. OSS contributors clearly stated that they were afraid that direct involvement of usability experts, especially in decision making, would overrule the democratic way of OSS, since it would be difficult to have a democratic debate against the only expert on the matter; "*Makes no sense to have 1 person deciding how the interface should look. I prefer the independent group approach. Fits better in the OSS model*". Another contributor described the possibility of contributions from usability experts as: "*It's a bit difficult. OSS people don't like too much to be told what to program. The human resources flow according to their personal interests, and maybe an usability expert by itself would not be sooooo useful*". However, OSS developers were positive towards external usability evaluations where the experts contributed with a usability report of the tested program: "*From time to time we get some usability reports from professional people. ... Once in a while they arrive and bless us with their wisdom. lol.*". Despite the appreciation of the knowledge of usability professionals and the usability reports, we sensed a gap between the technically minded contributors and those with a usability background. We consider it a paradox that OSS developers on one hand wanted a higher degree of usability in their software and furthermore appreciated the inspections made by external usability experts, yet the usability evaluation effort should ideally be performed by external persons or groups in the role of advisors.

We learned that in practice usability is not the first priority of all developers since most found technical challenges more interesting. Furthermore, there was reluctance to include usability experts in the decision making process because of a fear that the knowledge would make it difficult for others to influence the design of the user interface. OSS contributors were mainly positive towards usability but there was some confusion about exactly what the term covered. Cooperation between OSS developers and usability professionals is ideally performed with the usability people in the role as external advisers.

4.3 The OSS Development Process

OSS development reflects a babbling bazaar. We wanted to identify characteristics of OSS development methodology in order to compare the finding with a conventional software development paradigm. We identified four important factors of OSS development.

- Development in short iterative cycles
- Usability is an add-on property
- Democracy is important
- Trust is crucial

The development process is characterized by short iterative cycles. When we asked the developers about the development process in the OSS community we found that all developers independently thought that the development process was characterized by development in small iterative steps. Two of the developers compared the basics of OSS development to the extreme programming (XP) development paradigm [2]; "*I'm an XP (development methodology) fan, so I start doing things in short steps. I add functionality and elements to the interface as needed, but try to group them in a meaningful way*". XP is an unconventional development methodology with a strict focus on short development cycles, frequent testing and direct involvement of the customer. However, despite the apparent simplicity of XP the method relies on a set of procedures. To mention a few: Frequent meetings between developers and users, pair programming and extreme emphasis on tests. Hence, OSS development contains elements of XP but arguably others have been left out. Nevertheless, OSS is developed iteratively and does contain testing. This iterative method gives the opportunity to include usability evaluation as an element in each cycle. One of the interviewed OSS contributors stated "*I have to confess most of this stuff is not yet implemented*" about an external usability evaluation that was more than six months old. Therefore the short iterative cycles of the OSS development process need to be considered by usability professionals providing feedback to developers; feedback needs to be realistic and possible to implement within the current release cycle.

Usability is regarded as an add-on property. A typical understanding of usability was that it could be incorporated at a certain stage of the development process, for instance once the program could be compiled or had the desired functionality. The analysis showed that the developers had different ideas on when usability belonged in the development process. We identified four main stages (one respondent was not sure what to answer):

- In the beginning (12)
- Iteratively (5)

- In the end (1)
- During testing / QA (5)

This showed that there was no shared opinion about where in the development process the main usability effort should be made. Still, half of the respondents stated that it should be considered in the beginning of the development process. For instance one respondent stated that the usability effort should be "*At the beginning. Usability is harder to bolt on later, although it can be added later at the expense of creating a whole new interface*". There was difference of opinion even between developers contributing to the same OSS project. For instance contributors to Kopete independently stated that usability should be considered in four different stages. This showed that there was no general agreement on how usability should be incorporated in some projects. The low number of OSS contributors who answered that usability considerations should be an iterative process suggests that many considered usability as an add-on property. The usability professionals at Relevantive shared the impression that OSS developers often saw usability as an add-on property of the software and not as an integrated part of the development process.

Democracy is important in the OSS development process. Compared to traditional software development there was little formal leadership; "*OSS people don't like too much to be told what to program. The human resources flow according to their personal interests...*". As opposed to conventional software development we noticed that even though almost every OSS project had at least one project manager associated, this title did not equal leadership of the project. Often the title project manager reflected the person who founded the project rather than the person who kept track of everything or delegated tasks to other contributors. Still, statements were ambiguous "*I am the original author of Kopete. Kopete has no project manager. I am still the benevolent dictator. We have hardcore contributors, release dudes, etc but nobody manages the project*". On one hand he stated that there was no hierarchy with a designated project manager, on the other hand the particular project manager underlined his central role in the decision making process - even if the part about dictatorship was said in a humorous tone. Another of the interviewed persons stated "*Some persons, the 'fathers' of the project, have an outspoken voice and can persuade more easily about some issues*" In general, the project managers interviewed stated that one of the main concepts of OSS was the democratic way of developing software. This did not call for the project managers to make decisions concerning the project without involving the other project contributors. They stated that most major decisions were made democratically by everyone involved and discussed for instance on the mailing list.

Trust is crucial in the development process. Co-operation with the OSS community is based on trust and both developers and usability professionals contributing to OSS need to be prepared to build a rapport with other contributors. For instance Relevantive experienced that almost all problems faced when working with OSS developers were grounded in lack of trust, which made developers ignore suggestions from usability professionals. On the other hand the developers stated that the changes "*has to make sense, and we need to eval if it can be done. Some changes are too big to be done, not because of the idea itself, but because of the underlying code*". The nature of OSS, where contributors rarely meet in person, makes it necessary to judge others based on past merits. It can be difficult for a usability expert to display merits, since usability improvements are more difficult to measure than the programming of a new feature. Relevantive stated that when there was no face-to-face contact with the other person, the task of building trust could be the most strenuous task of co-operating with OSS developers - but none the less crucial for being heard. Relevantive found that attending various OSS conferences and gatherings was an excellent way to get acquainted with OSS developers and ultimately build the necessary level of trust. When trust had been established, the direct contact between usability professionals and developers resulted in a work environment which, in the view of Relevantive, was very gratifying.

We found it difficult to define the OSS development process in detail, and it may in fact be questioned whether a uniform 'OSS development model' exists. Interviewed OSS contributors agreed that the development model held similarities to methods like XP. They used short release cycles of the programs and implemented new features, without going through the long analysis and design stages of some conventional software development methodologies. Though frequent testing during the entire development process is a keystone of XP, we noticed that more than half of the contributors thought usability belonged in the beginning of development. Democracy is an ideal for OSS contributors in the way that discussions should be open and equal to everyone. The organization of work within OSS development depends highly on trust; contributors are measured on their previous merits. Overall we observed several factors that support Raymond's notion of the 'babbling bazaar' [20].

4.4 Usability Evaluation Methods in Practice

Common sense was the primary evaluation method. We wanted to investigate how usability evaluation fits into the OSS development process. Moreover, we wanted to identify some of the current usability methods adopted by OSS contributors. In the questionnaire survey 79 per cent of respondents answered that they followed common usability conventions and the same number of respondents stated that they used usability guidelines. Active usability evaluation was not used as frequently; 42 per cent answered that they used expert inspections, but they were rarely performed by usability professionals, 21 per cent mentioned a remote usability evaluation and about 8 per cent used a usability laboratory (Figure 4).

Guidelines replace usability evaluation. The methods most often used were formal or informal guidelines, and inspirations from similar programs. One set of guidelines often mentioned in the interviews were the KDE user interface guidelines. These guidelines define standards for menu layouts and user interface structure within the KDE and in addition it also provides a programming framework for application design. In the interviews the framework was highlighted as a usability enhancing factor since "*basically the only way to escape the guidelines is when you try to make a strange ui component, which is not part of the framework*". Nevertheless, the notion that this would ensure a high degree

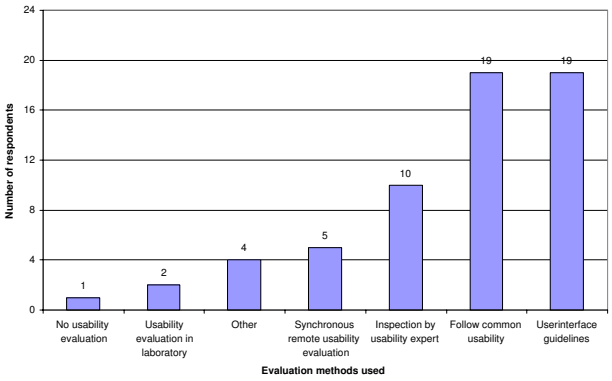

Figure 4: The developers were asked which evaluation methods were used in their project if any. The respondents could choose more than one method.

of usability was rejected by the interviewed usability professionals, of whom some had been deeply involved with the development of the KDE user interface guidelines. Though they considered the emergence of standards and consistency to be some of the most important parts of usability, they firmly stated that guidelines alone were not sufficient; *"People learn to use a program even if the design is stupid - so make standards even if they are poor ... guidelines are not the solution to all usability problems"*. Guidelines were generally good as they were helping to make standards in the look and feel of applications and provide the end user with a consistent experience. However, the usability professionals also said that "*guidelines can only be made for general items and not for specialized functions*" and they did not think guidelines could replace usability evaluation.

In regards to consistency an OSS developer mentioned that "*I have the opinion that even if some interface paradigm is not so ergonomic, it should always be consistent*". Another OSS developer had knowledge about some usability principles and used it in his own design of user interfaces: "*There are some conventions I personally use, like the 7 rule. It states that the human being can take in one shot only about seven elements on a given group. If it's more than 7, the brain starts to group the elements in smaller groups it can handle. So if you have an intuitive interface it will be better if you have less visible options*". He explained that he had never received formal usability training and that he had learned about usability by reading books and from internet based sources.

Though formal user interface guide lines were used and though some OSS developers showed knowledge about usability principles, we got the impression that most people just used common sense and inspiration from similar programs.

Money is a deciding factor. Only two of the respondents mentioned that they had used laboratories for usability evaluation. One of the interviewed developers thought that the economy of accessing a laboratory was a key issue: "*I think that the usability aspect is sort of harder for open source projects to do, so some sort of easier way or cheaper way to do this would actually be very welcoming, I would think, because you can't rent a decent lab*". Usability evaluations of OSS conducted by Relevantive were often performed as inspections by the professionals themselves. This testing method was not chosen because it was deemed the most usable, however, often it was the only possible test form. In most cases Relevantive did not receive any funding to finance their evaluation of OSS software. Hence, they were not able to pay 'real users' to participate in conventional usability testing in a dedicated laboratory, though they considered this a more effective evaluation method.

Remote usability evaluation is not a substitute for a laboratory evaluation. Five questionnaire respondents indicated that they used remote usability evaluation methods (Figure 4). When we investigated this approach to usability evaluation further during the interviews, we discovered that their idea of remote usability evaluation concerned troubleshooting of technical problems - for instance by connecting to a users computer and performing 'live' bug-fixing. The usability professionals at Relevantive were hesitant towards using remote methods where a conventional think aloud evaluation was performed remotely. In their opinion facial expressions were vital for identifying usability problems. A remote setup would also introduce too many problems, if the test users had to setup an environment consisting of web-cams, shared desktop and audio connections by themselves. A number of studies of remote usability testing mentioned the benefits of letting the test user operate in their normal working environment without the stress of a room full of recording equipment and observers [1, 5, 12, 23, 28]. At Relevantive they did acknowledge such advantages, however, they also thought that the current general level of usability was so low that the benefits of natural settings would not be exploited. They also thought that the users real environment would only add to the difficulty of performing the usability evaluation.

5. DISCUSSION

It remains to be seen whether the OSS community can provide user friendly interfaces to the same extent as conventional software development. During our research and in the empirical studies we noticed four common mantras about OSS and usability that we wish to challenge and discuss:

- OSS development is always democratic
- OSS will solve the 'software crisis'
- Usability problems are just bugs

5.1 OSS Development is Always Democratic

Is OSS democratic? We found that OSS projects in general had flat organizational structures and that OSS developers in our study praised the democratic organization of the development process. Furthermore, 25 per cent of the respondents in the questionnaire survey indicated that they contributed because of ideological reasons. The survey by Boston Consulting Group showed similar results [16]. Yet, you could argue whether the OSS development process is indeed democratic. Admittedly the development process and the flow of communication is open to anyone, however, the 'ideal' situation with no formal leader is not necessarily a sign of democracy. Raymond instead chose to describe the OSS development model as 'Meritocratic', indicating that

the previous contributions of the developers founded the basis of the social structure and influence on future decisions [20]. Trudelle stated that UI designers involved with OSS should be "*willing and able to engage the beast, for they can only get the needed leverage from impressing those doing the work - it's a meritocracy out there*" [29]. This was the exact point that our interview with both Relevantive and the OSS developers revealed; a level of trust must be built through merits in order to participate fully in the development process. One of the interviewed OSS contributors jokingly referred to himself as "*the benevolent dictator*" because he had initiated the project, and others supported the notion that there is a level of hierarchy among OSS contributors. However, these sociological patterns of OSS still need further research to be fully understood. For instance Feller and Fitzgerald called for further examinations that clarifies whether the ideals of collective public good, selfless behaviour, and absolute democracy are more than just an utopian illusion with no founding in the practical development process [7]. We found that democracy is one of the OSS ideals and there is indeed an open discussion among contributors; yet we also noticed that there are informal power structures in OSS development that are not necessarily founded in democracy.

5.2 OSS Will Solve the 'Software Crisis'

Will the OSS approach solve current software problems? The notion of a 'software crisis' has been evident during the last 30 years of computing. It covered the problems of delivering quality software that lived up to the requirements of the users, staying within budgets and deadlines [4]. In a famous paper, Brooks stated that "*as we look to the horizon of a decade hence, we see no silver bullet. There is no single development, in either technology or in management technique, that by itself promises even one order-of-magnitude improvement in productivity, in reliability, in simplicity*" [8]. Though methods for structured software development have evolved substantially, the media still report of failed software projects with the mentioned problems. It has been claimed that OSS development could possibly solve some of these problems [7]. The latest version of commercial systems like Microsoft Windows has a long release cycle, has been repeatedly delayed, and core features has been canceled during the development process. This suggests that the concept of 'cathedral building' is getting increasingly difficult as software complexity rises. Raymond's argument is that software is getting increasingly complex and that today operating systems and other large programs can not be "*built like cathedrals, carefully crafted by individual wizards or small bands of mages working in splendid isolation, with no beta to be released before its time*" [20]. The 'Bazaar' approach of OSS is the opposite but Raymond admits that it is not usable in all situations: "*It's fairly clear that one cannot code from the ground up in bazaar style. One can test, debug and improve in bazaar style, but it would be very hard to originate a project in bazaar mode. Linus didn't try it. I didn't either. Your nascent developer community needs to have something runnable and testable to play with*" [20]. Arguably, the OSS approach is not a full scale development model but rather an underlying philosophy with a set of principles to use during development. In the interviews OSS contributors compared OSS development to XP in regards to small development cycles and a well maintained prototype. Methods like XP has evolved as a reaction to shortcomings of conventional software development when it comes to rapid software development [2]. The bazaar approach of OSS combined with the emphasis on testing in XP has the potential to solve some of the problems in regard to delays and code quality, however imagining that it is the 'silver bullet' for all types of development projects would be wrong.

5.3 Usability Problems are just Bugs

Should usability problems be handled just like other 'bugs', and can the usability effort of OSS be handled within the current development framework? Wilson and Coyne suggest two fundamental approaches to track usability issues. One of them handles usability issues as regular bugs that are added to the bug database like any other problem, the other suggests a separate infrastructure to handle usability issues [31]. The latter approach has been taken by the usability professionals at Relevantive. They initiated the openusability.org project which goal was to bridge the gap between OSS developers and people with usability expertise. Openusability.org is a web-based on-line forum where it is possible to submit usability reports or request evaluations. According to Wilson and Coyne there are advantages and disadvantages to this separation from other bugs in the system. It puts usability out of the mainstream of development and makes it less visible to other developers. Furthermore, if usability issues are not in the general bug reports, they are less likely to be fixed. On the other hand they argue that comparing usability issues in the context of programming bugs is a risk, since a bug that makes the system crash obviously will get a high priority compared to a 'cosmetic' problem in the user interface. They also mention that current bug databases do not have sufficient categories to fully describe usability issues [31]. In our study we found that OSS developers prioritized an open and free negotiation process of potential changes to the project. The negotiation process within OSS has been the subject of a study by Sandusky and Gasser [24]. They examined negotiation in the context of the OSS bug report database Bugzilla and found that negotiation takes place in 61 per cent of bug reports and 27 per cent contains evidence of negotiation of several issues. The open negotiation process can sometimes cause problems making decisions. Trudelle described Netscape's experience about this in the Mozilla project. He found out that although the open discussion where everyone could participate was beneficial, sometimes it was more productive to create a 'by-invitation' group of UI owners and stake holders to settle issues and make authoritative decision, that had proved difficult to agree upon [29]. Hence, this experience should be considered by those developing systems for OSS usability collaboration; who has the authority to end an intractable argument and how will heated discussions be resolved?

We can not say for sure how usability problems are dealt with most effectively within OSS. However, we find it interesting to see the progress of current usability initiatives and the experience with these systems will tell, whether it works better than simply treating usability problems as a bug.

6. CONCLUSION

In this study we explored opinions about usability among developers and usability experts involved with OSS. Furthermore, we gathered information about the current practice of usability evaluation. The study included a questionnaire survey, interviews with OSS developers and interviews with usability professionals involved with OSS. This provided us with multiple perspectives on the subject of usability within OSS.

Overall we found that the developers were interested in usability but in practice most of the effort was based on common sense. They appreciated external usability evaluations performed by volunteer usability professionals, as long as they respected the decision making process. The usability professionals agreed that usability evaluation should be performed by external groups of experts.

Usability evaluation has not been the top priority of many OSS projects but attention to the subject is increasing among OSS developers. Moreover, there are several obstacles for usability work within OSS. Developers ought to have a basic understanding of usability theory in order to integrate usability considerations in the development process. Development is performed geographically distributed. Finally, the lack of resources and evaluation methods fitting into the OSS paradigm poses a problem.

We used mailing lists, forums and chat to contact potential respondents and we received a total of 24 answers to the questionnaire. Any reference in this article to quantitative data from the questionnaire should be considered indicative only. There is potential bias as two of the interviewed developers were associated with the same project. The interview with the usability professionals at Relevantive was also biased since all the interviewed people related to the same company.

7. ACKNOWLEDGMENTS

We would like to thank the staff at the Human Computer Interaction Research Unit at Aalborg University. We would especially like to thank our project counsellor Jan Stage for his invaluable feedback, suggestions and criticism. A special thank goes to Anders Rune Jensen, 'Taupter' and Duncan who participated in the interviews. Furthermore, we would like to thank Ellen Reitmayr, Björn Balazs and the rest of the staff at Relevantive for their co-operation and hospitality during our visit in Berlin. Finally, we would also like to thank all the participants in the questionnaire survey.

8. REFERENCES

[1] V. Bartek and D. Cheatham. experiences in remote usability evaluations. http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/50?OpenDocument&../Publish/1116/$File/paper1116.pdf, 2004.

[2] K. Beck. *Extreme programming explained: embrace change.* Addison-Wesley Longman Publishing Co., Inc., 2001.

[3] C. Benson, M. Muller-Prove, and J. Mzourek. Professional usability in open source projects: Gnome, openoffice.org, netbeans. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1083–1084, New York, NY, USA, 2004. ACM Press.

[4] E. W. Dijkstra. The humble programmer. *Commun. ACM*, 15(10):859–866, 1972.

[5] S. Dray and D. Siegel. Remote possibilities?: international usability testing at a distance. *interactions*, 11(2):10–17, 2004.

[6] S. Eklund, M. Feldman, M. Trombley, and R. Sinha. improving the usability of open source software: Usability testing of staroffice calc. http://www.sims.berkeley.edu/~sinha/opensource.html, 2001.

[7] J. Feller and B. Fitzgerald. A framework analysis of the open source software development paradigm. In *ICIS '00: Proceedings of the twenty first international conference on Information systems*, pages 58–69, Atlanta, GA, USA, 2000. Association for Information Systems.

[8] J. Frederick P. Brooks. No silver bullet: essence and accidents of software engineering. *Computer*, 20(4):10–19, 1987.

[9] N. Frishberg, A. M. Dirks, C. Benson, S. Nickell, and S. Smith. Getting to know you: open source development meets usability. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems*, pages 932–933, New York, NY, USA, 2002. ACM Press.

[10] E. Frøkjær, M. Hertzum, and K. Hornbæk. Measuring usability: are effectiveness, efficiency, and satisfaction really correlated? In *CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 345–352, New York, NY, USA, 2000. ACM Press.

[11] L. Gasser and G. Ripoche. Distributed collective practices and f/oss problem management. In *Conference on Cooperation, Innovation and Technologie*. CITE2003, 2003.

[12] D. Gough and H. Phillips. Remote online usability testing: Why, how, and when to use it. http://www.boxesandarrows.com/view/remote_online_usability_testing_why_how_and_when_to_use_it, 2003.

[13] O. S. Initiative. the bsd license. http://www.opensource.org/licenses/bsd-license.php, 2006.

[14] J. Johnson-Eilola. Open source basics: definitions, models, and questions. In *SIGDOC '02: Proceedings of the 20th annual international conference on Computer documentation*, pages 79–83, New York, NY, USA, 2002. ACM Press.

[15] S. Kvale. *Interview - En introduktion til det kvalitative forskningsinterview.* Hans Reitzels Forlag, 2001.

[16] K. Lakhani and B. Wolf.
bcg hacker survey.
Boston Consulting Group
http://www.ostg.com/bcg/BCGHACKERSURVEY-0.73.pdf,
2002.

[17] D. M. Nichols and M. B. Twidale. Usability and open source software. Technical Report 10/02, Department of Computer Science, University of Waikato, 2002. Working Paper Series ISSN 1170-487X.

[18] P. Nielsen.
Produktion af viden - en praktisk metodebog.
Teknisk forlag, 1998.

[19] S. Pemberton. Scratching someone else's itch: (why open source can't do usability). *interactions*, 11(1):72, 2004.

[20] E. Raymond. *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary.* O'Reilly, feb 1999.

[21] E. Raymond. *The revenge of the hackers.* O'Reilly and Associates, 1999.

[22] E. Raymond.
why open source will rule.
http://news.zdnet.com/2100-3513_22-871366.html,
2002.

[23] M. Safire.
remote moderated usability.
http://www.upassoc.org/usability_resources/-
conference/2004/im_safire.html,
2004.

[24] R. J. Sandusky and L. Gasser. Negotiation and the coordination of information and activity in distributed software problem management. In *GROUP '05: Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work*, pages 187–196, New York, NY, USA, 2005. ACM Press.

[25] R. J. Sandusky, L. Gasser, and G. Ripoche. Bug report networks: Varieties, strategies, and impacts in a f/oss development community. In *MSR 2004: International Workshop on Mining Software Repositories.* IEEE International Conference on Software Engineering, 2004.

[26] Smith, Engen, Mankoski, Frishberg, Pedersen, and Benson.
gnome usability study report.
http://developer.gnome.org/projects/gup/ut1_report/report_main.html,
2001.

[27] R. Stallman.
gnu general public license.
http://www.gnu.org/copyleft/gpl.html,
1991.

[28] K. E. Thompson, E. P. Rozanski, and A. R. Haake. Here, there, anywhere: remote usability testing that works. In *CITC5 '04: Proceedings of the 5th conference on Information technology education*, pages 132–137, New York, NY, USA, 2004. ACM Press.

[29] P. Trudelle. Shall we dance? - ten lessons learned from netscape's flirtation with open source ui development. In *CHI 2002.* Presented at the Open Source Meets Usability Workshop, Conference on Human Factors in Computer Systems, 2002.

[30] M. B. Twidale and D. M. Nichols. Exploring usability discussions in open source development. In *HICSS*, 2005.

[31] C. Wilson and K. P. Coyne. The whiteboard: Tracking usability issues: to bug or not to bug? *Interactions*, 8(3):15–19, 2001.

APPENDIX B

A Comparison of Remote Usability Evaluation Methods

A Comparison of Remote Usability Evaluation Methods

Morten Sieker Andreasen
Department of Computer Science
Aalborg University, Denmark
sieker@cs.aau.dk

Henrik Villemann Nielsen
Department of Computer Science
Aalborg University, Denmark
villeman@cs.aau.dk

Simon Ormholt Schrøder
Department of Computer Science
Aalborg University, Denmark
ormholt@cs.aau.dk

ABSTRACT

Based on an initial interest in the problems of usability evaluation within Open Source Software, this paper presents and discusses results from an empirical study of remote usability evaluation methods. The study includes a comparison of one synchronous and two asynchronous remote usability evaluation methods against an evaluation performed in a state-of-the-art usability laboratory. The aim of the comparison was to see whether the same number of usability problems was identified. Each of the four evaluation methods was performed with six users and the data analysis was performed using a highly structured procedure in order to minimize subjective bias. We found that the remote synchronous evaluation method using web-cams showed results almost identical to the laboratory evaluation. The asynchronous methods identified fewer problems than the other methods but the problems identified were mostly critical. Remote usability evaluation can provide results comparable to a laboratory usability evaluation.

Keywords

Usability, remote evaluation, empirical study.

1. INTRODUCTION

Remote usability evaluation is a term used when *"the evaluators are separated in space and/or time from users"* [8]. This definition states that there are two general types of evaluations; synchronous and asynchronous. When evaluating using synchronous methods the evaluator is separated from the user in space but not in time. On the other hand when conducting an asynchronous evaluation the evaluator is separated from the user in time and possibly in space.

Our motivation for researching remote usability methodology was founded in previous research about Open Source Software (OSS), where we examined how usability considerations were thought of and implemented in OSS projects from the perspective of developers as well as usability professionals [2]. This study showed that there was a genuine interest from both sides to increase the usability of OSS. It also showed that most usability evaluation methods used in the open source community were limited in scope. Guidelines and informal conventions were the primary elements of the usability effort and use of conventional usability evaluations was limited partly due to the lack of access to usability laboratories. The usability problems in many OSS user interfaces are well documented [6, 12, 13, 24, 26] and we found it interesting to investigate whether evaluation methods existed which were suitable for use in the context of OSS development.

OSS development is characterized by distributed collaboration between contributors to a specific project; a project can have hundreds of contributors spread worldwide [10, 23]. This makes it hard to employ conventional usability evaluation methods. Arguably methods suitable for the OSS community can with ease be performed by external usability evaluators. In this article we examine how three different approaches to remote usability evaluations compare to a laboratory evaluation in regard to the identification of usability problems.

Addressing the key question, this article describes an empirical study performed to investigate whether remote evaluation methods are comparable to conventional usability evaluation. In section 2 we describe previous research about remote usability evaluation and how this study contributes with new knowledge. In section 3 we present the methods of the various evaluation types while also describing the method used for data analysis. In section 4 we present the results of the empirical study, and in section 5 we discuss themes related to the research question based on overall findings and finally we conclude on the key question of this article in section 6.

2. RELATED WORK

The field of remote usability evaluation lacks a thorough methodical comparison of several methods. In existing literature we identified five methods whereof two were synchronous and three were asynchronous (Table 1). The majority of literature concerning remote usability evaluation simulates a conventional laboratory think aloud evaluation by using video and or audio connections plus remote desktop sharing. Especially the research carried out by Dray et al, Whitaker, and Hammontree et al described how this method has been used and what the advantages and disadvantages of the method were [11, 15, 19]. Some of the advantages found included cost efficiency, a potentially more diverse pool of suitable test users, and that the method identified the same problems as a think aloud evaluation performed in a laboratory - in some cases even more. Dray et al also pointed to problems connected to this evaluation method. In their study they found that it was difficult to build trust between moderator and user, it demanded longer setup time, and if there was a malfunction in the hardware or software it was very difficult to re-establish the test setup [11].

	Synchronous		Asynchronous		
	Usability evaluation	**Usability inspection**	**Self administered web study**	**Self report of critical incident**	**Logged use pattern**
Text communication	[3, 4, 5]	[22]		[16, 8]	
Questionnaire or multiple choice	[3, 4, 5, 19, 17, 31, 1]	[22]	[25]	[17]	[32, 29]
Workflow logging	[17]	[22]			[32, 29]
Screen shot (still image)	[15]	[22]			
Live observation	[28, 14, 3, 4, 5]				
Audio communication	[28, 14, 3, 4, 5, 19, 7, 17, 31, 15, 1, 11]	[9]		[17]	
Video capture of screen	[28, 14, 3, 4, 5, 19, 7, 17, 31, 15, 1, 11]	[9]		[16, 8, 17]	
Video capture of face	[19, 15, 11]				

Table 1: Table of remote usability testing methods identified in literature.

A few articles also describe experiments where a type of synchronous remote usability inspection was performed [22, 9]. Others like Hartson et al discussed strengths and weaknesses of several kinds of remote evaluation and presented two case studies [17]. Existing literature concentrate on listing pros and cons of the various methods when used under different circumstances. It lacks thorough descriptions of the used methods, the process of data analysis and often the empirical data is not provided. Olmsted et al performed a self administered web study, where the user filled out a questionnaire during the test, to a conventional usability study in a laboratory. The study revealed a number of disadvantages to this remote evaluation method as there was a low frequency of completion amongst the users, it was very time consuming, and it provided less qualitative information [25]. Hartson et al took this a step further and tried to make the users identify and report the critical incidents themselves. The users were taught how to do this with minimal training and generally the study showed that the users only missed few of the critical issues found through conventional evaluation. The method also proved to be both cost and labour efficient as much of the work was moved from the evaluators to the users [8, 17, 16].
Finally, articles by Scholtz and Winckler et al described the use of automatic logging of the use patterns of the test users in order to identify usability problems [29, 32]. They listed the same disadvantages as Olmsted et al when analyzing logged use patterns. The lack of accurate qualitative data made the analysis difficult and it proved to be a lot less efficient compared to conventional usability evaluation methods [32].

As a research area remote usability evaluation is still at it's dawn. A common denominator which is characteristic for the above studies is that they mainly compare one remote usability evaluation method to a conventional method. The articles are often vague when describing the method used for data analysis and also present limited data as backing for the conclusions made. The area lacks a thorough methodical comparison of several alternatives to usability evaluation. In this article we have performed a systematic comparison of three remote evaluation methods intended to fill this void in the exiting research.

3. METHOD

Following the methodologies presented in the related work we have chosen to compare three types of remote usability evaluation with a laboratory evaluation as benchmark.

- Laboratory evaluation (LAB)
- Remote synchronous evaluation (RS)
- Asynchronous expert evaluation (AE)
- Asynchronous user evaluation (AU)

The details concerning participants and the tested system are the same for all methods so these two are described in the following. Afterwards the specifics for each methodology are described in more detail.

Evaluation method	**Female**	**Male**	**Sum**
LAB evaluation	4 (26,3)	2 (21,5)	6 (24,7)
SR evaluation	2 (26,5)	4 (24,0)	6 (24,8)
AE evaluation	2 (26,5)	4 (26,8)	6 (26,0)
AU evaluation	2 (23,0)	4 (26,0)	6 (25,0)

Table 2: Gender representation in the four usability evaluations. The number in the parentheses denote the average age.

Participants: A total of 24 users, 14 male and 10 female, participated in the four different usability evaluations. They were all students at Aalborg University aged between 19 and 30 (mean=25.13 , SD=3) (Table 2). All participants had experience using a computer and the internet, and the six participants of the AE evaluation had furthermore received formal training in usability evaluations through their education. The 18 users who had not received usability evaluation training were randomly assigned as test subjects to one of the three remaining evaluation methods; LAB, RS, and AU.

The users received compensation for their involvement in form of snacks and beverages.

System: We chose to test the Mozilla Thunderbird 1.5 (Danish version) e-mail client in all four usability evaluations. We wanted to test a system within a domain - in this case e-mail - that was familiar to the participants. During the screening and selection of participants we made sure that none of them had experience using Thunderbird, however, we made sure that they had all used an e-mail client like Outlook or Netscape mail, so they were familiar with the basic concepts of an e-mail application.

3.1 Laboratory Evaluation

Setting: The LAB tests were based on the think aloud protocol described by Rubin [27] and were performed in a state-of-the-art usability laboratory. The test participants and the moderator performed the tests in a designated test room fitted with cameras (Figure 1, room A) with one-way mirrors that made it possible to observe the tests from the control room. Furthermore the operators in the control room (Figure 1, room C) were able to communicate with the moderator via an ear-piece. The moderator and the participants were both seated in front of the same PC. The role of the moderator was primarily to ensure that the participants thought aloud while performing a task, but also to proceed to a new task if the user got stuck. Prior to the tests we constructed nine different tasks which the users should complete during the tests. The tasks included sub-tasks such as: Create a new mail account, set up a spam filter, add a friend to your contact list, and add a label to an e-mail.

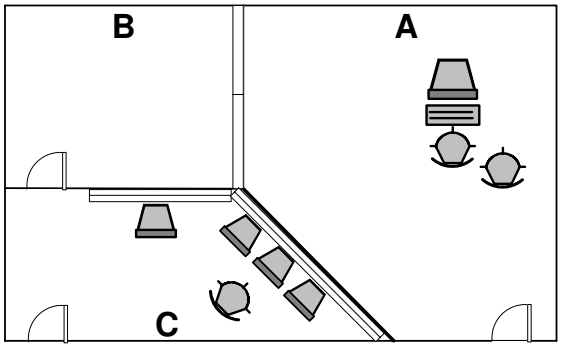

Figure 1: The setting used when conducting the LAB usability evaluation.

Procedure: Initially the moderator introduced the participants to the concept of a 'think aloud' evaluation. The moderator explained that the session was being recorded but emphasized that the recording would only be used for research, in order to make the participant feel relaxed and comfortable with the setup.
The participants were asked to solve the nine tasks. We did not specify a time limit, but encouraged the participants to try and solve all tasks without help from the moderator. After the sessions the participants were debriefed in a short interview about the evaluation method.

Data collection: In the LAB evaluation we recorded both audio and video feeds. The video feed consisted of the users desktop and a small video image of the test participants face in the bottom right corner of the screen.

3.2 Remote Usability Evaluation

Setting: The RS evaluation was based on the literature described in section 2. We learned that a simulated laboratory evaluation using tools such as web-cams, remote desktop connections and audio was the most frequently used and effective remote usability evaluation method. Hence, we designed a remote usability evaluation that took advantage of these tools. All though it was performed inside the usability laboratory, it was setup to simulate a remote evaluation environment. The moderator and the participants were in separate rooms (Figure 2, room A (moderator) and room B (participant)) and they could only communicate through an audio connection and web-cams (Figure 3). The one-way mirror between room A and B was covered with curtains. The operators in room C, the control room, were able to communicate with the moderator via an ear-piece (Figure 2, room C). We chose VNC and Microsoft Netmeeting as the testing platform since this software allowed shared desktop and communication via web-cams. We also chose to use Skype for the audio communication, since the audio quality through Netmeeting was not satisfactory.

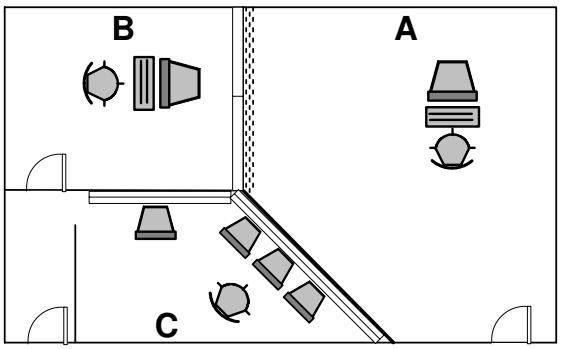

Figure 2: The setting used when conducting the RS usability evaluation.

Procedure: The procedure for the RS evaluation was equal to the LAB evaluation apart from the communication between the moderator and the participant.

Data collection: In the RS evaluation the recorded data consisted of the audio and video images that the moderator experienced through VNC, web-cam and Skype. This reflected a real world remote testing scenario where this would be all the information that the moderator received. The video feed consisted of the view of the users desktop provided by Netmeeting and a video image from the test participant's web-cam in the lower right corner. The test participant also had a web-cam image of the moderator in the lower right corner of the screen, however, this was not visible in the recorded material that was provided for further analysis.

Problem severity	Delay in task completion	Irritation	Expected action
Small	Less than 30 seconds delay	Slight irritation	Minor difference in expected action
Medium	More than 30 seconds delay	Average irritation	Significant difference in expected action
Large	Could not solve the task	High irritation	Critical difference in expected action

Table 3: This table provided the respondents with the information necessary to classify the problems.

Figure 3: A participant in the RS evaluation.

3.3 Asynchronous Usability Evaluations

Setting: The asynchronous evaluations were inspired by particularly one type of evaluation method identified in existing research about remote usability evaluation. Castillo et al and Hartson et al described positive experiences with a method called 'Self report of critical incident' [8, 17, 16]. In this evaluation method the participants not only performed the tests but also identified the main problems of the tested software, and thereby minimized the workload for the evaluators. We wanted to see if users without any formal usability knowledge were able to generate useful results. Thus, we chose to divide the method into two different parts; an asynchronous evaluation with experts of usability and an asynchronous evaluation with ordinary users. Both procedures were performed in remote locations at the participants own computers at a convenient time.

Procedure: Before the tests we made an installation manual to Thunderbird and put that on the first page of the online questionnaire. This was done in order to minimize the contact between the participants and the evaluator during or prior to the tests. The participants were presented with the same nine tasks as the participants of the RS and LAB tests. In this evaluation the tasks were an integrated part of an online questionnaire, where the users after each task could report any identified problems.

Data collection: The data collection for the asynchronous evaluations was done solely through the online questionnaire constructed through the UCCASS system. On top of guiding the participants through the assignments, the online questionnaire also gathered the input from the users in a MySQL database.
The questionnaire was designed in a way that made it possible for the participants to classify the identified problems. The problems could be classified as 'small', 'medium' and 'large'. These categories were correlated to the much used usability classifications 'cosmetic', 'serious', and 'critical'. The respondents were presented with a table specifying how to classify the specific problem (Table 3). Furthermore, it was possible to log where in the program they encountered a problem and how the problem intervened with the completion of the task.

3.4 Data analysis

In the four different usability evaluations we conducted a total of 24 tests with six participants in each. The data analysis of these tests was not commenced before all of the tests had been carried out. We were aware that there were some methodical challenges in the fact that we were only three usability evaluators to condense the problem lists from the empirical data. Thus, we constructed an analysis procedure that, according to Kjeldskov et al [21], minimized subjective bias of the evaluators.
The tests produced 24 objects for analysis; twelve usability evaluations performed in 'real time' and twelve asynchronous usability evaluations. The data from the LAB tests consisted of an audio recording and video captures of the participants face and desktop recorded by professional video equipment. The data from the RS evaluation consisted of a web-cam image and a video capture of the desktop and audio communication between moderator and the participants. The data from the asynchronous tests were semi structured problem lists, where the participant had already identified the problem areas and categorized them using Table 3.
The 24 sets of data were given a random identifier to minimize the subjective bias of knowing which test were being analyzed. Furthermore, each of the three evaluators randomly selected the order in which the sets of data were analyzed. In each set of data the evaluators identified the main usability problems and numbered them with a unique identifier to make it possible to trace each problem back to the original problem list. The evaluators then identified and categorized the problems found in each set of data. It was carefully noted in which set of data the problems were identified. This process took approximately 42 hours per evaluator, a total of 126 man-hours.
Hence, after the evaluation process a separate problem list of the tested software had been generated by each test evaluator. Ultimately the three test evaluators negotiated a complete problem list of the tested system until consensus had been reached (Table 14 on the last page). This process took approximately 30 man-hours. The final categorization of the problems were made by using 'worst case' categorization. This resulted in a problem being categorized as critical if just one evaluator had categorized it as such.

Following the effort to minimize subjective bias, we were interested in calculating the evaluator effect. This measure

relates the evaluators' individual performance to their collective performance [18]. Hertzum and Jacobson used the any-two agreement equation (Equation 1).

$$Avg.\ of\ \frac{|P_i \cap P_j|}{|P_i \cup P_j|}\ over\ all\ \frac{1}{2}n(n-1)\ pairs\ of\ evaluators \quad (1)$$

In the equation, P_i and P_j are the sets of problems detected by evaluator i and evaluator j and n is the number of evaluators. We calculated the evaluator effect using the any-two agreement equation in order to see how often evaluators agreed on identified usability problems. The average

	E1 E2	E1 E3	E2 E3	Avg.
Problems agreed on	29	30	28	29
Number of problems	42	45	43	43.3
Any-two agreement	69.0%	66.7%	65.9%	**66.9%**

Table 4: A calculation of the evaluator effect between evaluators E1, E2 and E3 using the any-two agreement formula (Equation 1).

percentage of any-two agreement in our data analysis was 66.9 per cent (Table 4). Hertzum and Jakobsen found that the average agreement between any two evaluators in twelve studies varied from 5 per cent to 65 per cent (avg. 22.4 %, SD=19.8) [18]. Compared to these figures we achieved a very high any-two agreement factor and this supports the validity of the usability problems identified and ultimately the final problem list.

In the analysis of the results we applied different statistical analysis tools. The total number of problems identified in the four setups, the number of critical, serious and cosmetic problems were analyzed through a Fisher's exact test. ANOVA one-way analysis of variance were used with the number of tasks solved, and with the time usage in the four evaluations. Furthermore, a Tukey post hoc comparison on the average number of problems identified in the evaluations was used.

4. RESULTS

Through four usability evaluations we attained knowledge about the effectiveness and usefulness of remote usability evaluation The results covered the following themes:

- Task completion
- Task completion time
- Usability problems identified
- Average number of identified problems
- Unique problems
- Opinions about the evaluation methods

4.1 Tasks Completion

The 24 test sessions resulted in an average of 8.1 out of 9 tasks completed. As seen in Table 5 the AE evaluation and the AU evaluation resulted in a higher number of solved tasks than the LAB evaluation and the RS evaluation. However, through an ANOVA test we found no significant difference in the number of tasks completed in the four evaluations (F[3,20]=0.68, p=0.575).

Evaluation method	Mean value	SD
LAB	8	1.1
RS	7.5	2.1
AE	8.5	0.8
AU	8.3	0.8
Total	8.1	1.3

Table 5: Tasks solved. SD= Standard deviation

4.2 Task Completion Time

The variation in completion time is not important. The results shown in Table 6 depicted that the participants in the two asynchronous evaluations, especially the participants in the AU evaluation (M=1:03:48, SD=0:48:37) had a longer task completion time than the participants in the LAB evaluation (M=00:22:10, SD=00:05:20) and the participants of the RS evaluation (M=00:22:30, SD=00:03:31). A significant difference was found when comparing the time used for the four evaluations through an ANOVA test (F[3, 20]=3.514, p=0.034). Yet, we found no pairwise differences between the evaluations using a Tukey's post hoc test, at a five per cent significance level.
In the AE and AU evaluations, however, the online questionnaire only recorded the begin and end time. Therefore we do not know if the participants had any breaks during the test sessions, and therefore we do not know the exact time spent on the test, which is an essential element when comparing AE, AU and LAB evaluations. For example, one of the participants in the AU evaluation used considerable more time (02:39:34) to complete the test, than the average for the rest of the asynchronous tests (00:54:39).

4.3 Usability Problems Identified

The number of problems identified in the evaluations vary widely. The 24 usability test sessions resulted in 46 usability problems (Table 14 on the last page). Based on the classification scheme in Table 3 we classified 24 of the 46 usability problems as critical, 10 as serious, and 12 as cosmetic.

Laboratory evaluation
In the LAB evaluation the evaluators identified 35 of the 46 usability problems: 22 of these problems were critical, 5 were serious, and 8 were cosmetic. The laboratory evaluation was performed as a benchmark to constitute a basis for comparison of test results.

Laboratory evaluation vs remote synchronous evaluation
In the results from the RS evaluation we saw that 38 of the 46 overall problems were identified. This corresponded well to the number of problems identified in the LAB evaluation, and according to a Fisher's exact test (Table 7) there was no significant difference in the number of problems identified in the two evaluations (p=0.6073). A similar pattern was found in the identification of critical problems. The LAB evaluation and RS evaluation both identified 22 of the 24 critical problems, where 4 of the 22 problems were found in only one of the two evaluations. Thus, 18 of the problems identified

Task completion time and number of identified problems	**LAB N=6**		**RS N=6**		**AE N=6**		**AU N=6**	
Task completion time (SD)	22:10 (05:20)		22:30 (03:31)		45:29 (18:51)		1:03:48 (48:37)	
	C	**%**	**C**	**%**	**C**	**%**	**C**	**%**
Critical (24)	22	92%	22	92%	15	63%	11	46%
Serious (10)	5	50%	8	80%	3	30%	2	20%
Cosmetic (12)	8	67%	8	67%	3	25%	0	0%
Total (46)	35	76%	38	83%	21	46%	13	28%

Table 6: Key results of the usability evaluations. In the table 'C' denotes the count of identified problems within the three categories of severity.

	LAB	**RS**	**AE**	**AU**
LAB		(p=0.6073)	p=0.0051 *	p<0.0001 ***
RS	(p=0.6073)		p=0.0004 **	p<0.0001 ***
AE	p=0.0051 *	p=0.0004 **		(p=0.1300)
AU	p<0.0001 ***	p<0.0001 ***	(p=0.1300)	

Table 7: Fisher's exact test for statistical significance. Calculated on the overall number of errors identified in the four evaluations. (p)=Not significant, *=Significant, **=Very significant, and *=extremely significant.**

in the two evaluations were the same. In the identification of serious and cosmetic problems the LAB evaluation and the RS evaluation attained almost equal results. The RS evaluation identified 8 of 10 serious problems and 8 of 12 cosmetic problems, which was slightly better than the LAB evaluation. In the identification of critical (p=1.000), serious (p=0.3498), and cosmetic (p=1.000) problems no significant difference was found through a Fisher's exact test.

Laboratory evaluation vs asynchronous expert evaluation
The AE evaluation identified a total of 21 of 46 problems, compared to the LAB evaluation that identified 35 of the 46 problems. A Fisher's exact test showed that there was a significant difference (p=0.0051) in the number of total problems identified in the two evaluations. In the identification of critical problems the difference between the LAB evaluation and the AE evaluation was not as distinct, since the AE evaluation identified 15 of 24 critical problems against the LAB evaluation's 22 of 24 critical problems identified. This showed that even though the AE evaluation did not find as many overall problems as the LAB evaluation the majority of the problems identified were critical. However, a Fisher's exact test still found a significant difference (p=0.0363) in the number of critical problems identified.
In the identification of serious and cosmetic problems the AE evaluation identified 3 of 10 serious and 3 of 12 cosmetic problems. The Fisher's exact test did not classify this as a significant difference in the number of serious (p=0.6499) or cosmetic (p=0.0995) problems identified, when comparing to the LAB evaluation.

Laboratory evaluation vs asynchronous user evaluation
The AU evaluation identified 13 of the 46 overall problems. A comparison of this result to the LAB evaluation through a Fisher's exact test found an extremely significant difference (p<0.0001) as seen in Table 7. In the identification of critical problems, the difference between the two evaluations was also classified as significant (p=0.0078), since the AU evaluation only identified 11 of the 24 overall critical problems where the LAB evaluation identified 22 of the 24 critical problems. This showed that the majority (84,6%) of the problems identified in the AU evaluation were critical. The difference in the number of serious problems identified was not classified as a significant difference according to the Fisher's exact test (p=0.3498). However, in the identification of cosmetic problems the AU evaluation did not find any problems, while the LAB evaluation identified 8 of the 12 overall cosmetic problems. In a Fisher exact test this resulted in a significant difference between the two evaluation methods (p=0.0013).

Asynchronous expert evaluation vs asynchronous user evaluation
We wanted to investigate the internal relation between the AE and AU evaluations since these were similar except for the usability knowledge of the participants. In Table 8 we classified the results of a Fisher's exact test comparing the critical, serious and cosmetic problems identified in the two asynchronous evaluations. This showed that there was no significant difference in the number of problems identified in the two asynchronous evaluation, despite the differences in problems identified.

Problems	**p**
Overall	0.1300
Critical	0.7702
Serious	1.0000
Cosmetic	0.2174

Table 8: Comparison of statistical significance between the AU and AE evaluations. P=significance level

4.4 Average Number of Identified Problems

The average number of problems identified per test session for the four setups varied immensely. In the results depicted in Table 9 we found that the average number of usability

problems identified by the test participants were almost the same in the RS and the LAB evaluations. Furthermore, the Tukey comparison did not find a significant difference in the number of problems identified ($p > 0.05$). However, we found a very significant difference when comparing the average number of problems identified by the participants of the LAB evaluation with the average number of problems identified by the participants of the AE and AU evaluations ($p \leq 0.001$).

	Mean	SD	Tukey comparison
Lab	15.33	4.41	
RS	16.67	2.42	$p > 0.05$
AE	4.67	2.66	$p \leq 0.001$
AU	3.17	1.72	$p \leq 0.001$

Table 9: Average number of problems solved. SD= Standard deviation. In the Tukey test the evaluations are compared to the LAB evaluation.

4.5 Unique Problems

Different usability evaluation methods often reveal unique problems. In our identification of unique problems we were inspired by the identification of action areas in Karat et al [20].

We identified the problems that were identified in one test session only, and the problems that were identified by only one evaluation method.

Problems identified in one test session

As shown in the Sum column in Table 10, none of the 24 critical problems, 2 of the 10 serious problems, and 6 of the 12 cosmetic problems were identified in one test session only. This underlines the validity of the critical problems, and furthermore shows that 50 per cent of the overall cosmetic problems were only identified in one test session.

Problems identified in one evaluation method

In the Sum column in Table 10 we saw that 1 of the 24 critical problems, 5 of the 10 serious problems, and 6 of the 12 cosmetic problems were identified in only one of the four evaluations. Thus 23 of the overall 24 critical problems were identified in more than one evaluation method. Furthermore, 50 per cent of the serious and cosmetic problems were only identified in one evaluation method.

Table 6 showed that the AE and AU evaluations identified 63 and 50 per cent of all critical problems, where the LAB and RS evaluations both identified 92 per cent of all critical problems. However, Table 10 shows that the asynchronous evaluations did not identify any unique critical problems. Therefore, we know that the critical problems identified in the AE and AU evaluations were the same problems as identified in the LAB and the RS evaluations.

Furthermore we found that the RS evaluation identified 1 critical, 3 serious, and 2 cosmetic problems that were not identified in any of the other evaluations, which was the largest amount of unique problems identified in all of the four evaluation methods.

4.6 Opinions about the Evaluation Methods

The degree of irritation was low in all evaluation methods. We wanted to know how test participants felt during the usability evaluation. Hence, we asked participants to grade their degree of irritation on a scale of five going from 'very low' to 'very high' and, furthermore, we collected qualitative feedback about the evaluation methods. Out of the 24 participants, 21 answered that their degree of irritation was 'low' or 'very low'. Participants in the LAB and RS evaluations mostly indicated 'very low' while most of the AE participants only chose 'low' (Figure 4). One of the participants in the RS evaluation even preferred this method to the conventional method; "*I liked this evaluation method better than the traditional method where the test leader looks over your shoulder.*". In both LAB and RS evaluations participants stated that it was awkward to 'think aloud'. Furthermore, one of the AE participants mentioned that switching between the program and the questionnaire was a source of frustration; "*The worst 'problem' was to move back and forth between the browser and the program while memorizing the task.*". Overall we found that there was no major difference between the LAB evaluation and the remote evaluation methods in regard to irritation of the test participant.

	Lab N=6	RS N=6	AE N=6	AU N=6	Sum N=24
Critical (24)	**0** (0)	**0** (1)	**0** (0)	**0** (0)	**0** (1)
Serious (10)	**1** (1)	**0** (3)	**0** (0)	**1** (1)	**2** (5)
Cosmetic (12)	**2** (2)	**2** (2)	**2** (2)	**0** (0)	**6** (6)
Total (46)	**3** (3)	**2** (6)	**2** (2)	**1** (1)	**8** (12)

Table 10: Identification of unique problems. The numbers outside parentheses are unique problems identified in only one test session. The number inside parentheses are problems identified by only one evaluation method.

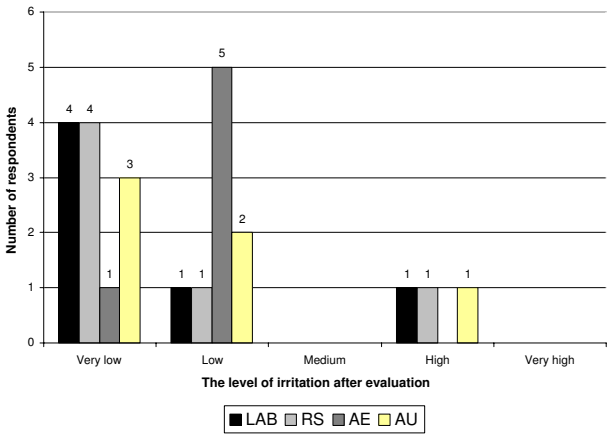

Figure 4: The degree of irritation among participants in the four different evaluation methods.

The questionnaire implementation was not optimal. One of the non-expert users commented that the reporting of problems was not entirely logical and the relatively low number of problem reported using the asynchronous methods (Table 6) suggests that this is a major problem of the questionnaire implementation. For instance respondents had the possibility of splitting a problem into three subproblems but this confused respondents. One participant did not under-

stand the structure of the questionnaire: "*I answered 'No' to finding problems most times but I still used the following possibility to comment where I clarified the small problems I experienced - this does not match with answering 'No' to the finding of problems.*". This was seen often in particularly the AU evaluations and it is a problem in respect to the asynchronous setup. Some participants did not understand the structure of the questionnaire and did not use the functionality as intended

We did not find any evidence that the methods performed remotely were perceived as more irritating than the LAB evaluation. However, some participants found it strange to think aloud which presents a problem for this method.
Some feedback suggested methodical problems connected to the setup of the asynchronous evaluations. These were the cause of problems since important data was possibly not reported. In future implementations of an asynchronous method, it is crucial for the evaluators to construct the questionnaire in a way that minimize the risk of misinterpretations.

5. DISCUSSION

We carried out this study in order to compare three methods for remote evaluation of usability against a LAB evaluation. In the previous section we presented the main results in order to answer our key question. Through the empirical research, the study also brought four other related questions to our attention, which are important to discuss:

- Is the physical presence of the moderator important?
- Can participants in AE and AU identify and classify problems?
- Which problems are not found by the asynchronous methods?
- Is the evaluator's choice of classification method important?

5.1 The Importance of the Moderator

Is the physical presence of the moderator important? The comparison between the LAB and RS evaluation methods showed that they identified the same number of usability problems and almost the same distribution of cosmetic, serious and critical problems (Table 6). The comparison also showed that there was no significant difference in either task completion time or the number of completed tasks between LAB and RS evaluation (Table 5). Thompson et al claimed that one of the disadvantages of a remote synchronous usability evaluation was increased task completion time [31] but we did not find anything to support this notion. Nor did we experience the 'distance' between the moderator and test user to be an influential factor which several sources suggested [1, 3, 4, 5, 11, 14, 28]. The similar results of the two evaluation methods in our study show that the physical presence of the moderator is not important. In fact, several test users expressed that the RS evaluation method was less stressful than the LAB test. One test user who had tried a laboratory evaluation earlier stated: "*I liked this evaluation method better than the traditional method where the test leader looks over your shoulder.*" and others supported this saying that the video image of the moderator was a positive element, since it was nice to be able to see the attitude of the moderator. In our study the RS evaluation method simulated a laboratory usability evaluation via web-cams, audio connection and shared desktop. We found that this helped communicate facial expressions and body language in a way that minimized the problems of remote usability evaluation. The main parameter that varied between the two evaluation methods was the physical presence of the moderator. In the LAB evaluation the moderator was sitting next to the test user, in the RS evaluation the test user and the moderator could see a web-cam image of each other in the corner of the screen. Overall we found that it is possible to compensate the physical absence of a moderator, and in our setup of a RS usability evaluation, this did not have any effect on the identified usability problems. The moderator's physical presence was not important.

5.2 Identification and Classification of Problems in the Asynchronous Evaluations

Can participants in AE and AU identify and classify problems? We have shown that the asynchronous methods did not identify as many problems as the LAB and RS evaluations. This was the focus of this papers key question. Even though the key question was not directed at finding out whether the users themselves were able to identify and classify problems, we chose to investigate this further.

Identification of Problems
The participants did not identify as many problems as the evaluators. One of the main ideas in the asynchronous methods is that the participants themselves identify the problems which makes the method very time efficient for the evaluators. We wanted to see how many problems the participants identified compared to the number of problems identified by the evaluators (Table 11). The evaluator's problem identification was made using the classifications and comments provided by the individual participants. In the table it is obvious that the evaluators identified more problems than the participants. On average the test participants identified 1,92 problems while the evaluators identified 4,17 problems per test. This shows that the evaluators identified more than twice as many problems as the participants. One possible explanation of this could be that the participants in the asynchronous evaluation solved most of the tasks (Table 5). It is possible that this affects the final judgment on whether or not the participants experienced problems, as they feel success after managing to solve the tasks even when encountering problems. This is supported by Hartson et al [17].

Classification of Problems
The participants classified the problems different than the evaluators. As discussed in the results (Table 6) the two asynchronous methods identified fewer problems than the two other methods. In addition, we noticed that the participants classified almost all problems as cosmetic and only 3 problems as critical (Table 12). When comparing the classifications made by the participants to the evaluator's classifications there is a notable difference. As an example participant T17 identified 3 problems and classified them all as cosmetic. The problems were in the final problem list classified as critical. This illustrates the difference in the classifications made.
Both expert and non expert users classified almost all prob-

	AE						AU					
	T14	**T18**	**T16**	**T13**	**T17**	**T15**	**T19**	**T20**	**T23**	**T24**	**T22**	**T21**
Severity	**Participants**											
Cosmetic	1		1	2	3			2	1	2	2	2
Serious	1					1	2					
Critical	1	1						1				
Severity	**Evaluators**											
Cosmetic	1	1		1								
Serious		1		2	1		1					
Critical	7	4	3	4	3	3	3	5	1	4	2	3

Table 11: A table showing the number of problems identified. The table shows how many problems the participants identified and how many problems the evaluators identified in the data provided by the participants.

	AE						AU					
	T14	**T18**	**T16**	**T13**	**T17**	**T15**	**T19**	**T20**	**T23**	**T24**	**T22**	**T21**
Severity	**Participants**											
Cosmetic	P8		P8	P13,39	P18,22,25			P32,41	P22	P10,22	P16,33	P3,14
Serious	P32					P22	P22,23					
Critical	P3	P22						P14				
Severity	**Evaluators**											
Cosmetic												
Serious				P39			P23					
Critical	P3,8,32	P22	P8	P13	P18,22,25	P22	P22	P14,32,41	P22	P10,22	P16,33	P3,14

Table 12: A table showing the classification of the problems found by the participants. The table shows how the participants classified the problems they found and how the evaluators classified the same problems. P refers to the problems in the final problem list (Table 14)

lems as cosmetic. A study by Skov and Stage showed that novice evaluators who had received a conceptual tool and an introduction to this tool were better at identifying and classifying usability problems than ordinary users [30]. Our study did not support this, since there was no obvious difference between expert users and ordinary users.

We have shown that the asynchronous methods did not identify or classify usability problems in the same way as the evaluators. However, we do not believe the method should be rejected. We have shown that the asynchronous method did not identify unique problems (Section 4.5), but the problems identified were mostly classified as critical. The problems identified are in-fact critical and they present real problems in the evaluated application.

5.3 The Asynchronous Methods Revisited

Which problems are not found by the asynchronous methods? We found that the LAB and RS evaluations identified almost twice as many problems as the AE and AU evaluations (Table 6). When investigating this difference closer we found that many of the problems *not* identified by the asynchronous evaluations were of a specific type.

The following list shows examples of problems which were not identified by the asynchronous evaluations:

- Cosmetic problems
  - Confusion about 'New...' button in the message filter (P20).
  - Incorrect keying when creating contact (P27).
  - Confusion about 'missing' accept-button after setting message filter (P15).
- Serious problems
  - Confusion about the difference between 'Spam filter' and 'Message filter' (P17).
  - Confusion about the term 'New mailing list' in contacts (P29).
  - Confusion about the difference between deleting an e-mail and moving it to trash (P39).
- Critical problems
  - Confusion about drop-downs concerning local folders or account name during setting of message filter (P9).
  - Confusion about 'Filter log' in the (P19) message filter.
  - Problem finding the 'Label' function (P43).

When examining these problem descriptions it is apparent that many of the problems involve confusion for the participants. These problems were identified when closely analyzing the video material, by interpreting mouse movements, facial expressions, and the comments made when solving the tasks. The problems are valid but it was uncertain whether the users were aware that they were encountering a problem. This observation is important since it is crucial for the asynchronous evaluations that the users recognize that they have encountered a problem. If the users themselves do not recognize that they are having a problem it will not be reported in the asynchronous methods. This finding is supported by Hartson et al [16] who found that the users had

Task completion time and number of identified problems	LAB N=6		RS N=6		AE N=6		AU N=6	
	C	%	C	%	C	%	C	%
Critical (12)	10	83%	12	100%	7	58%	6	50%
Serious (18)	13	72%	14	78%	8	44%	7	39%
Cosmetic (16)	12	75%	12	75%	6	38%	0	0%
Total (46)	35	76%	38	83%	21	46%	13	28%

Table 13: Overall usability problems categorized after a weight system (cosmetic=1, serious=2, critical=3). In the table 'C' denotes the count of identified problems within the three categories of severity.

to be prompted continuously to keep reporting the problems faced and that users were often slow to recognize that they were encountering a problem. However, in Table 5 we learned that the users of the asynchronous evaluations on average solved a higher number of tasks. This supported the notion that the participants of the asynchronous evaluations did not report as many problems as the participants of the LAB evaluation.
It is evident that there is a difference in the number of problems identified by the individual evaluation methods. There is, however, one thing the problems which were not identified have in common. The type of problems missed were typically problems which entailed confusion. We suggest that the users are not always aware that they are encountering a problem.

5.4 Choice of Classification Method

Is the evaluator's choice of classification method important? In the analysis of the empirical data we used a 'worst case' principle. When the three evaluators merged their problem lists, a problem would be categorized as critical if just one evaluator had found it to be so. We wanted to investigate whether this choice of categorization made a large impact on the results. We tried to perform alternative calculations based on a system where cosmetic, serious and critical problems were given a weight from 1 to 3 (Table 13). The change of categorization method resulted in a decrease in the sum of critical problems from 24 to 12. The total number of serious problems increased from 10 to 18 and cosmetic problems went from 12 to 16. Using the worst case classification, a Fisher exact test showed only a significant difference in the number of critical problems identified in AE and AU compared to LAB (p=0.0363 and p=0.0078). This significant difference in the number of critical problems identified was not present when using the weight based system (p=0.3707 and p=0.1930). Except for this change there were no major differences between the two methods of categorization and the internal relations between the four evaluation methods remained largely the same.

6. CONCLUSION

In this article we examined how three different approaches to remote usability evaluation compared to a conventional laboratory think aloud usability evaluation. We set up a remote synchronous evaluation using web-cams, audio connection and shared desktop and an asynchronous questionnaire based evaluation performed by both users with and without usability training. In the asynchronous evaluations the users identified the usability problems themselves. The empirical study and data analysis was performed following a strict structure in order to minimize subjective bias from the evaluators. The study found that a RS evaluation was comparable to a LAB evaluation. They identified almost the same number of usability problems. In addition, test users spent the same time completing the evaluation.
The AE and AU evaluation methods identified fewer problems than the LAB evaluation. The classification by the test users did not match the classification of the evaluators, however, almost all reported problems turned out to be critical. There was no significant difference between users with and without usability knowledge.
Remote usability evaluation is a usable alternative to conventional methods. We suggest that remote methods can be considered in situations where there is no access to a laboratory or in a domain where it is not possible to use conventional methods.

Each evaluation was performed with six users. The method for data analysis with randomization was used in order to minimize subjective bias. The RS evaluation was performed in a laboratory simulating the distance between the test user and the moderator. A real world setup would reveal a number of technical problems or performance issues that we have chosen to ignore. The AE and AU evaluation methods depended on our actual implementation in a questionnaire based system.

It would be interesting to perform remote usability evaluations outside the controlled environment of a usability laboratory. This would highlight some of the practical problems connected to remote usability evaluation. Furthermore, it would be interesting to establish co-operation with one or several OSS projects in order to let the OSS contributors evaluate the methods.

7. ACKNOWLEDGMENTS

We would like to thank our project counsellor Jan Stage who contributed with constructive criticism and advice when needed. We would also like to thank Mikael Skov for his help and advice regarding statistical calculations and the procedure for data analysis. Finally, we would like to send a special thank to the 24 participants of the usability evaluations.

8. REFERENCES

[1] M. Ames.
final report on remote usability studies.
http://www.ocf.berkeley.edu/~morganya/research/dmp/report.html, 2005.

[2] M. S. Andreasen, H. V. Nielsen, and S. O. Schrøder.

usability in open source software development: Opinions and practice.
http://www.cs.aau.dk/~sieker,
June 2006.

[3] V. Bartek and D. Cheatham.
experience remote usability testing, part 1.
www-106.ibm.com/developerworks/library/wa-rmusts1/,
January 2003.

[4] V. Bartek and D. Cheatham.
experience remote usability testing, part 2.
www-106.ibm.com/developerworks/web/library/wa-rmusts2.html,
February 2003.

[5] V. Bartek and D. Cheatham.
experiences in remote usability evaluations.
http://www-3.ibm.com/ibm/easy/eou_ext.nsf/Publish/50?OpenDocument&../Publish/1116/$File/paper1116.pdf,
2004.

[6] C. Benson, M. Muller-Prove, and J. Mzourek. Professional usability in open source projects: Gnome, openoffice.org, netbeans. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1083–1084, New York, NY, USA, 2004. ACM Press.

[7] A. B. Brush, M. Ames, and J. Davis. A comparison of synchronous remote and local usability studies for an expert interface. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems*, pages 1179–1182, New York, NY, USA, 2004. ACM Press.

[8] J. C. Castillo, H. R. Hartson, and D. Hix. Remote usability evaluation: can users report their own critical incidents? In *CHI '98: CHI 98 conference summary on Human factors in computing systems*, pages 253–254, New York, NY, USA, 1998. ACM Press.

[9] G.-J. de Vreede, A. Fruhling, and A. Chakrapani. A repeatable collaboration process for usability testing. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 1*, page 46, Washington, DC, USA, 2005. IEEE Computer Society.

[10] B. J. Dempsey, D. Weiss, P. Jones, and J. Greenberg. Who is an open source software developer? *Commun. ACM*, 45(2):67–72, 2002.

[11] S. Dray and D. Siegel. Remote possibilities?: international usability testing at a distance. *interactions*, 11(2):10–17, 2004.

[12] S. Eklund, M. Feldman, M. Trombley, and R. Sinha. improving the usability of open source software: Usability testing of staroffice calc.
http://www.sims.berkeley.edu/~sinha/opensource.html,
2001.

[13] N. Frishberg, A. M. Dirks, C. Benson, S. Nickell, and S. Smith. Getting to know you: open source development meets usability. In *CHI '02: CHI '02 extended abstracts on Human factors in computing systems*, pages 932–933, New York, NY, USA, 2002. ACM Press.

[14] D. Gough and H. Phillips. Remote online usability testing: Why, how, and when to use it.
http://www.boxesandarrows.com/view/remote_online_usability_testing_why_how_and_when_to_use_it,
2003.

[15] M. Hammontree, P. Weiler, and N. Nayak. Remote usability testing. *interactions*, 1(3):21–25, 1994.

[16] H. R. Hartson and J. C. Castillo. Remote evaluation for post-deployment usability improvement. In *AVI '98: Proceedings of the working conference on Advanced visual interfaces*, pages 22–29, New York, NY, USA, 1998. ACM Press.

[17] H. R. Hartson, J. C. Castillo, J. Kelso, and W. C. Neale. Remote evaluation: the network as an extension of the usability laboratory. In *CHI '96: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 228–235, New York, NY, USA, 1996. ACM Press.

[18] M. Hertzum and N. Ebbe Jacobsen.
the evaluator effect: A chilling fact about usability evaluation methods.
International Journal of Human-Computer Interaction, 13, 2001.

[19] J. Houck-Whitaker.
remote testing versus lab testing.
http://boltpeters.com/articles/versus.html,
2005.

[20] C.-M. Karat, R. Campbell, and T. Fiegel. Comparison of empirical testing and walkthrough methods in user interface evaluation. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 397–404, New York, NY, USA, 1992. ACM Press.

[21] J. Kjeldskov, M. B. Skov, and J. Stage. Does time heal : a longitudinal study of usability. In *OZCHI '05: Proceedings of the 19th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction*, pages 1–10, Narrabundah, Australia, Australia, 2005. Computer-Human Interaction Special Interest Group (CHISIG) of Australia.

[22] F. S. H. Krauss. Methodology for remote usability activities: A case study. *IBM Systems Journal*, 42(4):582–593, 2003.

[23] J. Y. Moon and L. Sproull.
essence of distributed work: The case of the linux kernel.
http://www.firstmonday.org/issues/issue5_11/moon/index.html,
2000.

[24] D. M. Nichols and M. B. Twidale. Usability and open source software. Technical Report 10/02, Department of Computer Science, University of Waikato, 2002. Working Paper Series ISSN 1170-487X.

[25] E. Olmsted and M. Gill.
in-person usability study compared with self-administered web (remote-different time-place) study: Does mode of study produce similar results?
UPA2005, 2005.

[26] E. Raymond. *The revenge of the hackers*. O'Reilly and Associates, 1999.

[27] J. Rubin.
Handbook of Usability Testing.
Katherine Schowalter, 1994.

[28] M. Safire.
remote moderated usability.
http://www.upassoc.org/usability_resources/-conference/2004/im_safire.html,
2004.

[29] J. Scholtz.
adaption of traditional usability testing methods for remote testing.
IEEE, 2001. Proceedings of the 34th Hawaii International Conference on System Sciences.

[30] M. B. Skov and J. Stage. Supporting problem identification in usability evaluations. In *OZCHI '05: Proceedings of the 19th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction*, pages 1–9, Narrabundah, Australia, Australia, 2005. Computer-Human Interaction Special Interest Group (CHISIG) of Australia.

[31] K. E. Thompson, E. P. Rozanski, and A. R. Haake. Here, there, anywhere: remote usability testing that works. In *CITC5 '04: Proceedings of the 5th conference on Information technology education*, pages 132–137, New York, NY, USA, 2004. ACM Press.

[32] M. A. A. Winckler, C. M. D. S. Freitas, and J. V. de Lima. Usability remote evaluation for www. In *CHI '00: CHI '00 extended abstracts on Human factors in computing systems*, pages 131–132, New York, NY, USA, 2000. ACM Press.

Pid	Problem description	Severity
	Account settings	
P1	Problem finding the 'Create account' function	Critical
P2	Problems receiving new e-mails	Cosmetic
P3	Problem changing the account settings	Critical
P4	Confusion about the term 'Global inbox' when creating account	Serious
P5	Confusion about not entering a password when creating account	Serious
P6	Problem finding the account settings function	Critical
P7	Confusion about the functionality when creating account	Cosmetic
	Message filter	
P8	Problem finding the 'Message filter' function	Critical
P9	Confusion about drop-downs concerning local folders or account name during setting of message filter	Critical
P10	Confusion about inactivity of the 'Run filter' button	Critical
P11	Uncertainty about the 'Run filter' functionality	Cosmetic
P12	General confusion concerning the message filter window	Critical
P13	Unexpected result when running the message filter	Critical
P14	Confusion during setting of filtering rules or destination folder	Critical
P15	Confusion about missing accept button after setting message filter	Cosmetic
P16	Problem finding the 'Run filter' function	Critical
P17	Confusion about the difference between 'Spam filter' and 'Message filter'	Serious
P18	Confusion about '+' when setting a message filter	Critical
P19	Confusion about 'Filter log' in the message filter	Critical
P20	Confusion about 'New...' button in the message filter	Cosmetic
	Contacts	
P21	Problem finding the 'Create contact' function	Critical
P22	Problem creating a contact based on a received e-mail	Critical
P23	No automatic input of first and last name when a contact is created based on a received e-mail	Serious
P24	Confusion about the meaning of 'Recipients preferred format'	Cosmetic
P25	Uncertainty about the division of information in contacts	Critical
P26	Missing cursor when entering address during creation of contact	Serious
P27	Incorrect keying when creating contact	Cosmetic
P28	Confusion about the meaning of 'New card' in contacts	Cosmetic
P29	Confusion about the term 'New mailing list' in contacts	Serious
P30	Confusion about the layout of the contacts	Cosmetic
	Spam filter	
P31	Confusion about dialog box concerning the spam filter	Serious
P32	Confusion about the functionality of the spam filter	Critical
P33	Problem finding the 'Spam filter' function	Critical
P34	Confusion about the two drop-downs when configuring the spam filter	Critical
P35	Confusion about the drop-down used to select account in the spam filter	Critical
P36	Confusion about the difference between 'Delete mail' and 'Delete mail after X days'	Critical
P37	Problem marking an e-mail as spam	Critical
P38	Confusion that marking an e-mail as spam does not start the 'Spam filter' function	Critical
P39	Confusion about the difference between deleting an e-mail and moving it to trash	Serious
	General	
P40	Confusion about counting number of mails in folders	Cosmetic
P41	Confusion on how to create a folder	Critical
P42	Problem showing folders using the '+'	Cosmetic
P43	Problem finding the 'Label' function	Critical
P44	Confusion about the result of right and left clicking an e-mail	Serious
P45	Double clicking the Inbox results in opening a new instance of the Inbox - the user do not notice this	Cosmetic
P46	Confusion about drop-down when creating a new folder	Serious

Table 14: The final problem list.

APPENDIX C

Data Analysis Procedure in Paper 2

In the second research paper we conducted twenty-four usability tests spread across four different evaluation methods:

- Laboratory evaluation (LAB evaluation)
- Remote synchronous evaluation (RS evaluation)
- Asynchronous expert evaluation (AE evaluation)
- Asynchronous user evaluation (AU evaluation)

To compensate for the fact, that we were only three evaluators to condense problem lists from the twenty-four sets of empirical data, we constructed a procedure for data analysis inspired by Kjeldskov et al [22] in order to minimize subjective bias. The procedure made sure that each set of data was analyzed by each evaluator in random order. Figure C.1 has been subdivided into six phases corresponding to the following paragraphs.

Phase A The four evaluations produced 24 objects for analysis named T1 to T24. The results of the LAB evaluation, T1 to T6, consisted of a video capture of the participants face, desktop and and audio recorded by professional equipment in a state-of-the-art usability laboratory.

The results of the RS evaluation, T7-T12 included a capture of web-cam image of the participants face, and a video capture of the desktop using remote desktop software. Furthermore, audio communication between moderator and participant was recorded through Internet telephony software.

The results of the AE (T12-T18) and the AU evaluations (T19-T24), resulted in semi structured problem lists, where the participants had already identified and categorized the problem areas.

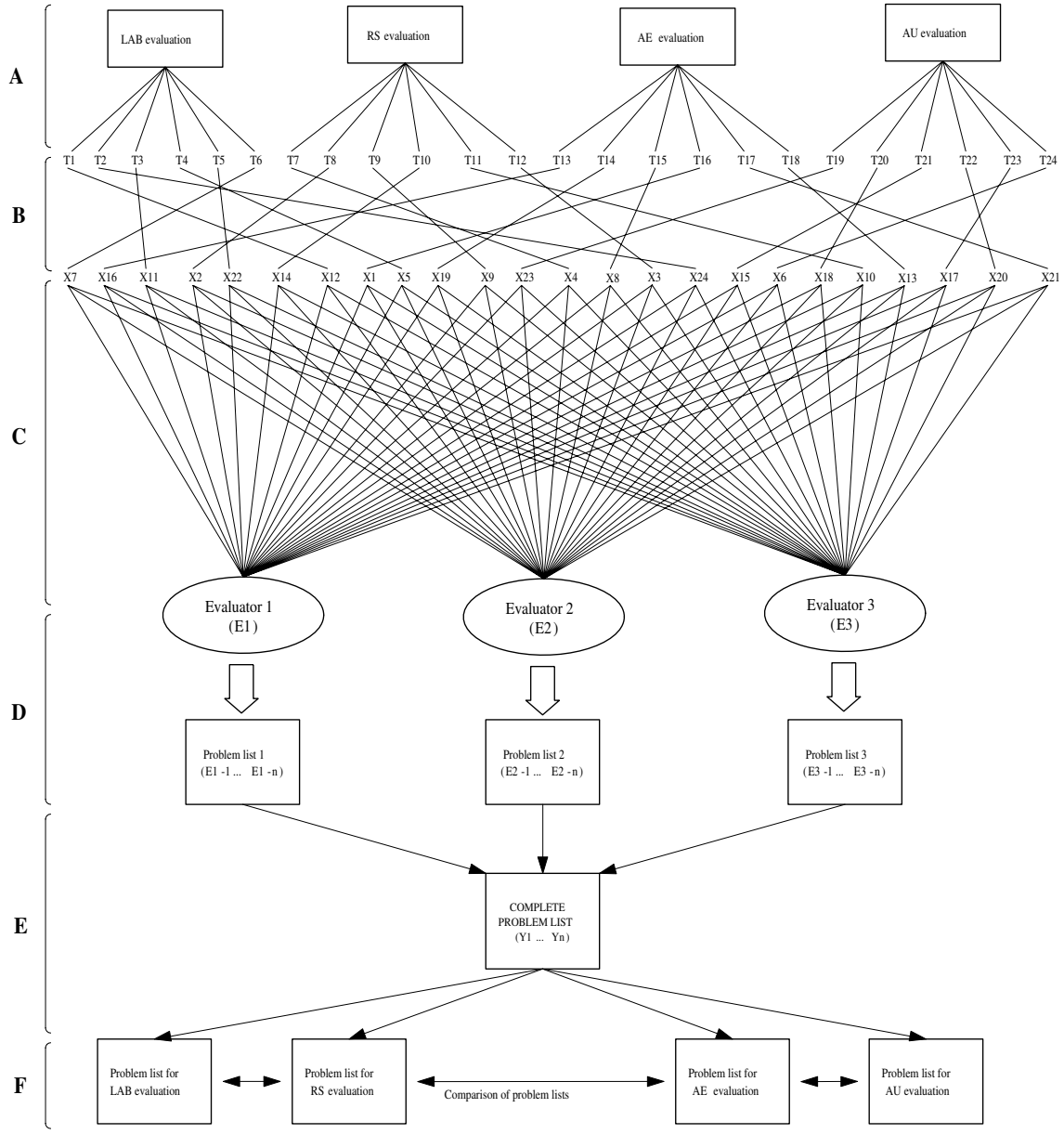

Figure C.1: Analysis procedure

Phase B The 24 sets of data were given a random name from X1 to X24, to minimize the bias of knowing exactly which test was being analyzed. Thereafter, each of the three test evaluators (E1, E2, E3) randomly selected the order that they would generate problem lists for each set of data. The evaluators identified the main usability problems in each set of data and numbered them with a unique identifier.
For example: **E1-X3-1** *(Evaluator 1 - test X3 - problem number 1)*

Phase C The evaluators then categorized the problems found in each set of data into overall usability problem named E1-1, E1-2, E1-n and so forth. It was carefully noted in which set of data the problems were identified.
For instance: **E1-3: E1-X3-1, E1-X7-5, E1-X22-3**

(Evaluator 1 - overall problem 3 - consisting of problems E1-X3-1, E1-X7-5 and E1-X22-3)

Phase D The evaluators constructed an overall problem list each, so that three separate problem lists of the tested software existed.

Phase E Ultimately the three test evaluators negotiated a complete problem list of the tested system until consensus had been reached. The result was another list of problems from Y1 to Yn. For each problem Yn it was listed which 'sub problems' it consisted of.
For instance: **Y4: E1-3, E1-8, E2-3, E2-9, E3-1, E3-8**
(Problem Y4 - consisting of overall problems E1-3, E1-8, E2-3, E2-9, E3-1, E3-8)

Phase F After the merge of the problem list was complete, it was possible to 'trace back' each problem Yn to its origin Tn.