

T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ

MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

**Açık Kaynak Kodlu Yazılım Projelerinin
Kullanılabilirliğe Yönelimlerinin Ölçümü
İçin Bir Metrik Model Geliştirilmesi**

GÖRKEM ÇETİN

DOKTORA TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

GEBZE

2012

T.C.

GEBZE YÜKSEK TEKNOLOJİ ENSTİTÜSÜ

MÜHENDİSLİK VE FEN BİLİMLERİ ENSTİTÜSÜ

Açık Kaynak Kodlu Yazılım Projelerinin

Kullanılabilirliğe Yönelimlerinin Ölçümü

İçin Bir Metrik Model Geliştirilmesi

GÖRKEM ÇETİN

DOKTORA TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

GEBZE

2012

DANIŞMAN

YARD. DOÇ. DR. MEHMET GÖKTÜRK



DOKTORA TEZİ JÜRİ ONAY SAYFASI

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
...16.01.2012... tarih ve 2011/02 sayılı kararıyla oluşturulan jüri tarafından 17.01/2012
tarihinde tez savunma sınavı yapılan Gökhan Çetisi.....'ın tez
çalışması Bilgisayarlı Hükümetçilik Anabilim Dalında DOKTORA tezi olarak kabul
edilmiştir.

JÜRİ

ÜYE

(TEZ DANIŞMANI) :

Yrd. Doç. Dr. Mehmet Bektürk
Gebze Yüksek Teknoloji Enstitüsü
Elektronik Mühendisliği Bölümü

ÜYE

:

Yrd. Doç. Dr. F. Erdem Sevilgen
Gebze Yüksek Teknoloji Enstitüsü
Elektronik Mühendisliği Bölümü

ÜYE

:

Yrd. Doç. Dr. H. Fatih Uzunelaz

ÜYE

:

Doç. Dr. Hacı Ali Mantar
Gebze Yüksek Teknoloji Enstitüsü
Elektronik Mühendisliği Bölümü

ÜYE

:

Doç. Dr. Erkan Zengeroglu
Gebze Yüksek Teknoloji Enstitüsü
Elektronik Mühendisliği Bölümü

ONAY

G.Y.T.E. Mühendislik ve Fen Bilimleri Enstitüsü Yönetim Kurulu'nun
...../...../20... tarih ve sayılı kararı.

İMZA/MÜHÜR

ÖZET

TEZİN BAŞLIĞI: AÇIK KAYNAK KODLU YAZILIM PROJELERİNİN KULLANILABİLİRLİĞE YÖNELİMLERİNİN ÖLÇÜMÜ İÇİN BİR METRİK MODEL GELİŞTİRİLMESİ

YAZAR ADI: GÖRKEM ÇETİN

Özgür ve açık kaynak kodlu (Ö/AKK) yazılım geliştiricileri, uzun bir süre boyunca kullanıcı merkezli bir tasarımı takip etmek yerine yazılımların özelliklerinin öne çıktığı, sonuç olarak ürünün kullanılabilirliğinin göz ardı edildiği bir geliştirme yöntemi kullanagelmışlerdir. Bunun arkasında pek çok neden olsa da, ana sebebi geliştiricilerin kullanılabilirlik kavramına yönelik bilinç eksikliği ve proje sorumlularının İnsan Bilgisayar Etkileşimi (İBE) alanındaki çalışmalarının görece azlığıdır. Ö/AKK yazılımların kullanılabilirliğini artırmak için geliştiricilerin kendi projelerindeki kullanılabilirlik aktivitelerini ölçmeleri ve uygun kullanıcı merkezli tasarım yöntemlerini kullanmaları gerekmektedir.

Bu tez, anket çalışması, literatür taraması ve Ö/AKK yazılım projelerinin incelenmesi sonucunda tanımlanan metriklerle bir ölçüm modeli ortaya koymaktadır. Ö/AKK yazılımların kullanılabilirlik seviyelerini artırmaya ve İBE uzmanları ile geliştiricilerin birlikte çalışma yöntemlerini iyileştirmeye yönelik olan bu çalışmada, bir ölçüm yöntemi önerilmiş ve geçerliliği örneklerle denenmiştir. Bunun yanında, ölçüm sürecini kolaylaştırmak için bir uygulama geliştirilmiş ve tanımlanan metriklere karşılık gelen değerlerin her bir projeye göre hesaplanmasına imkan verilmiştir.

SUMMARY

**TOPIC: A MEASUREMENT BASED FRAMEWORK FOR ASSESSMENT
OF USABILITY READINESS OF OPEN SOURCE SOFTWARE
PROJECTS**

AUTHOR: GÖRKEM ÇETİN

For a long time, free and open source software (F/OSS) developers tend to build feature-centric projects rather than following a user centred design, ignoring the necessity of usability inside the resulting product. While there are many reasons behind this, main cause can be stated as the lack of awareness of usability from developers' point of view and little interaction of project stakeholders with Human-Computer Interaction (HCI) studies. In order to increase the usability hence the competitiveness of open source software applications, developers need to measure their usability activities within the project, therefore conduct appropriate user centered design methodologies.

This thesis is concerned with defining a measurement model and related set of metrics derived from surveys and analysis of F/OSS projects, in order to evaluate the degree of usability readiness of a F/OSS project. Based on the motivation to leverage and improve the usability of applications and increase the efficiency of communication and collaboration level of HCI experts and developers, a measurement method is proposed and validated. Moreover, in order to streamline the measurement process, an application is developed for entering the data of criteria for a specific set of metrics and calculate the overall score of the usability matrix defined.

ACKNOWLEDGEMENTS

It would not have been possible to write this doctoral thesis without the help of the kind people around me. I'd like to thank all the people who've continuously supported me during the last few years. First of all, I'd like to thank to my advisor, Assoc. Prof. Dr. Mehmet Gokturk, for his dedicated patience, understanding and support during the whole process.

I'd also like to thank to my wife, Filiz, for her continuous support. Without her, it'd have certainly been much harder for me to finish my thesis. She has made available her support in a number of ways.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the project, including all the professors in Gebze Institute of Technology and kind and helpful assistants in the same university.

LIST OF ABBREVIATIONS

İBE	: İnsan Bilgisayar Etkileşimi
F/OSS	: Free & Open Source Software
HCI	: Human Computer Interaction
IEEE	: Institute of Electrical and Electronics Engineers
ISO	: International Standards Organization
GNU	: GNU is Not Unix
GPL	: General Public License
tOSSad	: Towards Open Source Software Adoption and Dissemination
COTS	: Commercial Off The Shelf
SQM	: Software Quality Management
GQM	: Goal Question Metric
QFD	: Quality Function Deployment
MUSiC	: Metrics for Usability Standards in Computing model
UI	: User Interface
QUIM	: Quality in Use Integrated Measurement
SUS	: System Usability Scale
ANSI	: American National Standards Institute
UMOS	: Usability Measurement of Open Source Projects
PUMA	: Project Usability Metrics Aggregator
HTML	: Hypertext Markup Language
XML	: Extended Markup Language

INDEX OF TABLES

<u>Table</u>	<u>Page</u>
Table 1: Usability as in ISO 9241-11	23
Table 2: List of well known open source applications	28
Table 3: Analysis of 20 years of F/OSS desktop	34
Table 4: KDE project bugs analysis	36
Table 5: Usability methods and models in distributed and closed groups	39
Table 6: Error reduction methods used in user survey	60
Table 7: Developers according to their job status	62
Table 8: Use of media for usability discussions	63
Table 9: Choice of evaluation method. Source: Analysis of survey data.	65
Table 10: Recommended usability metric table for distributed and open source software projects	79
Table 11: Symbols used to illustrate the effect on definition of the indicator	81
Table 12: Project usability metric criteria suggested by literature review	81
Table 13: KDE project usability metrics	86
Table 14: OpenOffice.org project usability metrics	88

INDEX OF FIGURES

<u>Figure</u>	<u>Page</u>
Figure 1: Inter-dependencies of main parts of the thesis work	18
Figure 2: Categories of software	29
Figure 3: Onion-like structure of F/OSS developers	30
Figure 4: Sample tasks of potential usability experts and contributors	32
Figure 5: Software Quality Characteristics (Boehm, Brown & Lipow, 1976)	43
Figure 6: GQM model hierarchical structure	44
Figure 7: Main screenshot of editor showing the introduction page	55
Figure 8: Percentage of bug trackers used	63
Figure 9: Percentage of focus group	64
Figure 10: Projects projects having plans for usability methodologies	66
Figure 11: Percentage of developers spending their time improving UI	66
Figure 12:User centric design process	73

TABLE OF CONTENTS

ÖZET	4
SUMMARY	5
ACKNOWLEDGEMENTS	6
LIST OF ABBREVIATIONS	7
1. Introduction	12
1.1. Background to the research	12
1.2. Research problems and hypothesis	13
1.3. Justification for the research	14
1.4. Methodology	18
1.5. Contribution to the research	19
1.6. Outline of the dissertation	21
2. Literature review	22
2.1. Introduction	22
2.2. Definition of usability	22
2.3. Usability processes	24
2.4. Open source software	25
2.5. Open source and proprietary software	28
2.6. F/OSS and usability	30
2.7. Software measurement models	39
2.8. Measurement frameworks	41
2.9. Current usability measurement models	45
2.10. Conclusion	48
3. Methodology	49
3.1. Introduction	49
3.2. Justification for the paradigm	49
3.3. Research procedure	51
3.4. Project usability metrics aggregator	54
3.5. Survey planning	56
3.6. Aim and motivation of the survey	57
3.7. Structure of the survey	59
3.8. Error reduction in survey	59
4. Analysis of data	62
4.1. Patterns of data	62
4.2. Interpretation of metric data (part 1)	71
4.2.1 User centered design	71
4.2.2 Developer base	72
4.2.3 Bug tracking	74
4.2.4 Project web page	76
4.2.5 Tasks and guidelines	77
4.2.6 Accessibility, internationalization and localization	78
4.3. Derivation of a metric model (part 2)	79
4.4. Final metric model table (part 3)	81
4.5. Validating the model	84
5. Conclusions and implications	90
5.1. Introduction	90

	11
5.2. Conclusions about research problem	90
5.3. Limitations and issues	92
5.4. Further research	94
6. APPENDIX 1: SURVEY QUESTIONS	96
REFERENCES	116
ÖZGEÇMİŞ	132

1. Introduction

This chapter gives an introduction by providing a background of the research and describing the research problem. We first introduce the basic underlying principles of the work and justify the research and its contribution. The chapter ends with the methodology, scope and the limitations of the research. Last section gives the outline of the dissertation.

1.1. Background to the research

Over the last 20 years, we see an abundance of literature on how usability can be used to leverage the return of investment (Nielsen, 1993b), increase end user satisfaction and lower the training budget (Karat, 1993). Setting interface standards, providing usability requirements before a software project begins, conducting heuristic evaluations and usability tests have been applied in order to evaluate and improve an application or a series of web pages to increase the performance of the end user.

On the other hand, little attention has been paid to usability of end user applications in F/OSS projects, although it is widely acknowledged as a success factor of desktop applications developed on an open medium, such as internet. Lack of common usability design guidelines and methods of communication between usability experts and F/OSS developers resulted in F/OSS software with relatively low level of usability. While there are counterexamples of this, it's worth investigating main factors to trigger lack of F/OSS application usability and concentrate on how to improve the user centric design experience of voluntary projects.

Different usability norms developed by proprietary vendors and F/OSS community need to be investigated. It should be noted that due to the commercial nature of proprietary software vendors, usability was (and still is) seen as a revenue generator: the more usable the software, the more licenses sold and the more revenues generated (Çetin, Verzulli, Frings, 2007). On the other hand, research reveals that main driving factor for F/OSS development is motivation of project stakeholders (Ye & Kishida,

2003; Lakhani & Wolf, 2003; Hertel, Niedner & Herrman, 2003), and usability has not been seen as the primary objective within the project aims. Moreover, main driver for developing a F/OSS product is the freedom of the movement itself, in which, this freedom does not enforce, imply or mean that the resulting application will be highly usable.

Under the light of this information, it can be concluded that there's a need to understand the basic blocks of how usability is taken care of in F/OSS projects. On the other hand, a measurement method and corresponding metric sets to assess the usability of F/OSS projects is need to help project stakeholders to self-evaluate their projects, which will sustain the current trend in desktop F/OSS development and influence the development of future F/OSS applications.

1.2. Research problems and hypothesis

In this thesis, the research problem is defined as “how can we formulate a measurement model describing the metrics of a distributed project process with a focus on usability?”. This thesis involves a detailed interpretation of two different domains: a) F/OSS development scheme, b) user centred design and usability. Under the light of the aim of the thesis that explore how open source development can benefit from distributed usability methodologies during the design process of an asynchronous project, main research question in this thesis is as follows:

RQ: How can we formulate the design and process activity for a collaborative project in order to benefit from usability and user centered design?

The hypothesis of the thesis can be defined as:

Hypothesis 1. We can derive or adopt a suitable measurement model for software projects developed online in order to assess the usability readiness of the project.

Hypothesis 2. The usability metrics we present within this model can be

used to measure the degree of usability of the project.

After having a thorough research, we find out that there's little information about how to apply usability metrics or interpret the scores from specific quality metrics for distributed projects.

Main aim of the thesis is to measure the usability maturity and usability readiness of distributed projects. There are two main reasons for describing a measurement based framework in order to assess the usability of distributed software projects:

1. A group of projects can be measured and outputs can be analyzed, yielding a comparison among the projects.
2. This analysis can be used in order to make amendments in project usability approach and suggest quality improvements in the project progress.

A measurement framework does not only define the metrics, but also guidelines on how to interpret the collected information. This allows researchers to have a basic understanding and interpretation of what different numbers mean within each metric category. This way, developers and quality assurance managers can basically make modifications on the project progress and process by interpreting the results.

1.3. Justification for the research

ISO 9241-11 (ISO, 1998) defines usability as the "effectiveness, efficiency and satisfaction with which a specified set of users can achieve a specified set of tasks in a particular environment". As a non-functional requirement, usability cannot be directly measured but quantified by means of different evaluation methods, e.g cognitive walkthrough, heuristic evaluation, paper prototyping and usability testing. Usability can also be used as a set of approaches while creating products, leading to reducing costs and increase the user satisfaction, by including people in the design stage who are directly the users of that particular product. Benefits of usability include increased

productivity include decreased training and support costs, increased sales and revenues, reduced development time and maintenance costs and increased customer satisfaction (Lund, 1997; Bloomer, 1997).

Free/Open Source Software (F/OSS) is a generic term which is used for software developed under a license which allows the use, modification and distribution of the software without claiming a fee for it. There are other terms, like “open source”, which describe a slightly different version of the F/OSS term, however the basic idea is same in terms of usage, distribution and modification. A surprising fact with F/OSS development is that while this model does not pose an in-line methodology with traditional software development models, there's a growing interest in F/OSS products showing an effective, rapid and reliable software development process. As of this writing, there are more than 1.5 million registered users in Sourceforge with more than 150.000 projects and Apache is the clear market leader with 53% market share according to Netcraft statistics.

As such, F/OSS usability is an important phenomenon deserving a separate study for itself. User centric design has not been the first priority of open source projects developed by people geographically distributed in all parts of the world. Thus, it can be argued that usability awareness and representation has been neglected for a long time. Traditionally, F/OSS has been successful in server products (Apache, MySQL, etc), libraries and compilers (GNU C compiler, Glibc, Qt, GTK) and console based applications (bash shell, pine, etc). These applications were used by experienced people in specific organizations like universities and research institutes and those who has used them demanded less training and usability criteria. Most of the users of these applications are relatively technically sophisticated and the average desktop user is using standard commercial proprietary software (Lerner and Tirole, 2002).

The lack of clear and well defined usability requirements, awareness of user centered design and a social collaborative tool to discuss usability issues

usually result in a poor evaluation for F/OSS products. This is a problem in ergonomic design and interface psychology, and hackers have historically been poor at it (Raymond, 1999). As F/OSS matures and enterprises start using F/OSS, it entered the mainstream area and enterprises started to demand not only reliability, security and efficiency, but usability, ergonomics and ease of use. As a result, in order to answer the demand from customers, some open source projects have tried to adopt techniques from previous proprietary work, such as explicit user interface guidelines for application developers (Benson, Adam, Nickell & Robertson, 2002).

Software evolution is defined as “change in the essential properties of a software product family over time” (Svetinovic, 2005). Normally, one expects the usability increases with evolution, however adding features also increases the complexity of the product. On the other hand, according to Nielsen, if product features doubles in number, then its complexity quadruples (Nielsen, 2008). So, as the software evolves, more features are added, and complexity is increased. This results in the fact that development process should be more inclined to spend time in ensuring usability of the interface.

On the other hand, one can't improve what is not measured. Literature surveys show that no metrics have been developed based on the priorities of the project plan, which in turn provide metrics that developers would want to watch for the health of their distributed project. If metrics were collected, then processes could be designed to collect information relevant to these metrics reduce it to numerical form for storage, display, and analysis (Balanced Scorecard, 2007). Considerable work has been done on the metrics level for general purpose software (Fenton, 1997), however there's a lack of quality work in the literature dealing with open source project metrics, and understanding the inner working principles of open source projects and gluing metrics is not analyzed in a deep manner. Moreover, lack of requirements document in many F/OSS projects hinders the necessity of usability requirements (Scacchi, 2002).

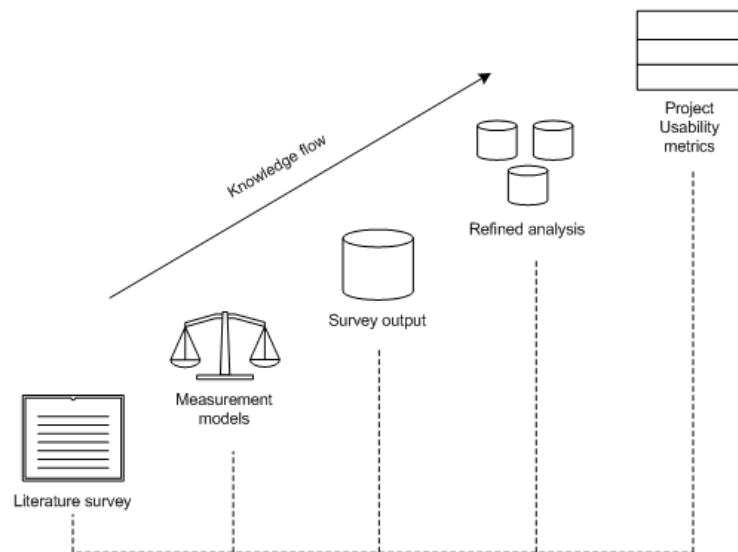
On the other hand, there's still a scarcity of published usability studies and test reports of F/OSS from academy. We are aware of studies on GNOME (Smith, 2002), Greenstone (Nichols, Thomson, Kirsten & Yeates, 2001), file browser screening test (Reitmayr, 2007), Linux desktop out of box experience (Göktürk & Çetin, 2007), Ganttproject usability test (Zhao, 2007) and OpenOffice.org usability test (Çetin, Verzulli & Frings, 2006), to name a few. While there are few formal researches on F/OSS usability, developers are well aware that F/OSS usability is a problem that should be solved and usability is a significant issue of unsuccessful F/OSS projects. According to Muehling (2004), "it becomes clear that F/OSS projects have a fundamental problem: they lack usability resources that help achieve better usable software for non-geek users". It's almost evident that lack of formal usability evaluation mechanisms result in lack of efficient usability assessment and reporting cases.

With the aforementioned objective and main research question, work to do in this thesis is threefold:

1. Analyzing how usability experts and F/OSS developers work, by investigating the involvement of usability experts in a distributed development environment.
2. Surveying usability experts and F/OSS developers in order to understand the dynamics and nature of usability discussions in F/OSS projects.
3. Understanding and examining different measurement models in software engineering and presenting a metrics model which will be used to estimate the usability readiness scale of a distributed F/OSS project.

These three work items can be stated within a more structured figure, which depicts the steps involved during the research (Figure 1).

Figure 1: Inter-dependencies of main parts of the thesis work



1.4. Methodology

The methodologies defined below are assumed to be sufficient to answer the research question “How can we analyze and formulate the design and process activity for a collaborative project in order to benefit from usability and user centered design?”, because we both used a qualitative and quantitative method to collect data.

In order to answer the research question, following series of actions have been deployed.

- a) Surveying (Chapter 3.5) will be used to collect quantitative data from HCI experts and F/OSS developers.
- b) Online collaboration environments are investigated in order to gather qualitative data (Chapters 2.6 and 2.9).
- c) Software measurement models are examined to investigate previous literature work in software engineering domain, in order to conclude on an appropriate metric model.
- d) A metric model will be developed (Chapters 4.1 – 4.4) to measure the usability readiness level of collaborative projects.

Finally, taking advantage of a, b and c; a general conclusion will be drawn.

As a side research, this thesis will investigate 1) the existing evidence of the usability of F/OSS. 2) outline the ways in which the characteristics of F/OSS development influence the software, and 3) describe how existing HCI techniques can be used to leverage distributed networked communities to address issues of usability.

This thesis has the potential aim of harvesting a fairly big amount of data since F/OSS projects offer a huge repository of online data via bug reporting systems, mailing lists, blogs and forums. Within the literature review (Chapter 2), a general investigation of distributed projects will be carried out, effectively helping out forming the metrics model defined in Chapter 4.

1.5. Contribution to the research

Until recently, usability has been defined in closed projects and groups, building a set of applications and applying relevant UCD methods during the development process. As discussed in Section 2 (Literature review), this approach is changing due to introduction of open, incremental development scheme of online projects with a fast prototyping and evolving nature. Most of the time, such organizations lack a coherent assessment of usability methodology which will lead to a usable end product, decreasing the usability level of the application.

Therefore, there's a growing need to define the basic characteristics, gaps and issues of usability processes in F/OSS projects and then provide a means of amending the issues referenced in the literature. This research first identifies these points by providing an extensive survey whose focus is F/OSS developers and HCI experts. Findings from this survey is categorized and listed as a resource input to derive the metric model, together with the help of literature survey. Accordingly, this thesis contributes to the field of open source, identifying gaps of usability in open source projects.

This research focuses on developing a metric model by examining the

usability processes of open source software projects running on the internet by collaborative development which is possible by a distributed and lean communication media. By conducting a research on this subject, a method and a corresponding software application is provided in order to measure the usability readiness of a project. Using the outcomes of this model, stakeholders will be able to increase the experience of user centered design process within their open source projects, by merely using the proposed metric table and applying the results. Therefore, we can say that this thesis contributes to the field of metrics and measurement domain, by implementing a metric model which can be validated through a series of tests.

After developing a metric model, a software editor is developed in order to ease the submission of real values gathered from a project. Each value can be in the form of nominal, ordinal, interval or ratio scale. This editor can be used for those who are not experienced with data entry process and complex mathematical foundations behind the framework. The main idea behind developing an editor is to assist F/OSS project coordinator to define, enter and calculate the metric values of a particular project. Therefore, it can be concluded that this thesis contributes to software engineering domain, by implementing an intuitive application to ease the data entry process.

Moreover, in order to manage F/OSS projects effectively, utilizing a user centric design process, development teams can use the results of this thesis to improve the usability performance of their projects, by constantly monitoring the project status with the help of the measurement metric given in Chapter 4, and provide relevant amendments to project structure.

1.6. Outline of the dissertation

The thesis will have a 5-chapter generic structure. Initial planning of the chapters are as follows¹ which has been adopted after examining a wide range of PhD theses structure.

¹ Implemented from “Structured approach to presenting PhD thesis”, retrieved from http://www.elec.uq.edu.au/doc/Thesis_guide/phdth2.html

1. Introduction: This chapter outlines the broad field of study and then leads into the focus of the research problem, by defining the core idea of the research related to open source, measurement models and usability. Within this chapter, the importance of research problem is justified by giving examples from literature. Finally an introductory overview of the methodology which will be used to derive the measurement model is explained.

2. Literature review: This chapter builds a theoretical foundation on which the research is based on. A comprehensive review of literature on the field of open source, usability and software measurement with a special attention to theories between measurement models and HCI has been carried out.

3. Methodology: This chapter describes the major methodologies used to derive the metric model in Chapter 4.

4. Analysis: This chapter presents patterns of online survey and literature survey results and analyses them for their relevance to research question.

5. Conclusion: The conclusion part includes the outcomes of the methodology used and analysis derived from the data collected. Moreover, concluding chapter suggests various areas for short and long term work and suggests how to reach the goals defined.

Appendix A: This section includes all the survey questions asked to individuals, together with the number of survey respondents and their answers.

References: A list of all references in the thesis work are depicted and sorted by last name.

2. Literature review

2.1. Introduction

This chapter presents a literature review of open source software, usability, software measurement and assessment of usability. We first study the definition of usability, which is explained in section 2.2. Then we examine open source software in detail, with a focus on project management and online collaboration. Next, we examine various methods of how usability is measured and assessed with a focus on validity. Finally we examine the concepts of software measurement and its role in software engineering domain, with a state of the art in measurement frameworks in use today covering usability and software engineering.

2.2. Definition of usability

The term “usability” has been defined in standards context in different ways. Likewise, since this term has a long history in parallel with software engineering, we see many definitions in literature from researchers. We first explain usability as it's defined in the literature below, except ISO 9241-11 which is a series of standards of ergonomic requirements for office work with display terminals which is already mentioned in Chapter 1.

- In ISO/IEC 9126-1 (1991) standard, usability is defined as “a set of attributes that bear on the effort needed for use and on the individual assessment of such use, by a stated or implied set of users”. Some countries, such as Japan, have adopted this as a national standard (Abran et al., 2003).
- IEEE 610.12-1990 standard defines usability as “the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component” (IEEE, 1990).

As it can be derived from the definitions from standards above, usability could be defined from a product-oriented (ISO 9126) or user-oriented (IEEE

610.12-1990, ISO 9241-11) view. Likewise, the definition of usability by the researchers derive from a number of views, therefore show different characteristics reflecting the domain context.

Eason (1988) defines usability as “the degree to which users are able to use the system with the skills, knowledge, stereotypes and experience they can bring to bear”. Shackel, Shneiderman, Nielsen and Preece also define usability from different points of view. Schackel (1991) proposes the usability of a system or equipment as “the capability in human functional terms to be use easily and effectively by the specified range of users, given specified training and user support, to fulfill the specified range of tasks, within the specified range of environmental scenarios”. According to Schackel, for a system to be usable, 4 criteria (effectiveness, learnability, flexibility and attitude) must be achieved. Nielsen (1993b) characterizes five attributes of usability as “learnability, effectiveness, error tolerance, satisfaction and memorability”. On the other hand, Shneiderman gives a similar item list but calls this definition “five measurable human factors central to evaluation of human factors goals” instead of usability. Given various definitions of usability, several researches tried to define the similarities and differences of these definitions. The following table, adopted from Welie, et al (1999), tries to sketch out the similarities between the definition of usability and measurable human factors from different perspectives.

Table 1: Usability as in ISO 9241-11

ISO 9241-11	Shneiderman	Nielsen
Efficiency	Speed of performance	Efficiency
	Time to learn	Learnability
Effectiveness	Retention over time	Memorability
	Rate of errors by users	Errors/safety
Satisfaction	Subjective satisfaction	Satisfaction

2.3. Usability processes

Usability of computer applications is becoming more important with the increase of introduction of relatively uneducated people with less skills to cope with the complexity of user interfaces. Lately, it has been observed that users are unwilling to put up with uncomfortable or difficult since experience with some current interfaces has shown them that software can indeed be easy to learn and pleasant to use (Nielsen, 1994). Therefore, high usability is a must requirement of a software application. However, without proper usability evaluation methods, it's not possible to measure the usability level of a particular application.

There are many usability evaluation methods for different domains, from mobile devices (Kjeldskov and Stage, 2004) to interactive TV sets (Pemberton and Griffiths, 2003; Daly-Jones, 2002; Gauntlett and Hill, 1999). These include expert evaluations like cognitive walkthroughs (Lewis, Polson, Wharton and Rieman, 1990; Wharton, Bradford, Jeffries and Franzke, 1992) or heuristic evaluations (Nielsen and Molich, 1990). We can classify usability evaluation methods into four, namely formal methods, automated methods, empirical methods and heuristics. A formal method (Card, Moran & Newell, 2003) benefits from formal and metric models to take usability measurements. Automatic method measures usability by measuring a user specification through an application. It stems from usability measures to describe the functioning of a system through the assessment of the software (Byrne et. al., 1994). An empirical method provides the most reliable methodology among all, since it incorporates a sample of real world users by examining the properties of usability and observing how a sizable sample of real users uses the system (Butler, 1996). The inspection method is based on the heuristics and rules of functioning, and it includes a single assessor who carries out the inspection of the application (Nielsen, 1990). Other sub-types of evaluation methods include verbal protocols (Ericsson and Simon, 1984), critical incident reporting (del Galdo, Williges, Williges and Wixon, 1987) and user satisfaction ratings (Chin, Diehl and Norman, 1988).

Usability processes produce competing products, by consistently evaluating the user interface using above-mentioned methods. Obviously, coordinated and cooperative design has been an important task lately, with the introduction of internet and internet-enabled software houses. In an open source world, there's a lack of understanding of "representative user", who will help assess the end product usability. Without this pool of users, developers can only imagine and develop according to their mental model, potentially ignoring end user tasks. According to Nielsen (1994), users should be involved in the design process through regular meetings between designers and users however this is often not possible when the software is developed internationally, speaking of thousands of kilometers between developers' country borders. Current attempts to gather developers and usability experts in regular meetings do not disperse the issue of "lack of interaction" with end users.

User contribution to product usability has been investigated in the literature for a long time. They are not designers, however can propose different interaction alternatives like designers when they are given enough time, support and environment. On the other hand, since reactive and critics to design could act as a barrier towards new incentives and building blocks for innovation. A communication channel is required, exploiting the advantages of internet, by guiding users to a productive discussion. In an internet-supported environment where software is developed in a free and unrestricted manner, a centralized, decision making authority is needed to focus the discussions not only for development, but also for usability merits. Both usability experts and end users expect to be encouraged to design smaller instances of a big picture so that these parts are regarded as design representatives or instances of illustration. During the discussions, a category-specific, or product-specific user interface guideline could also be produced.

2.4. Open source software

Free and Open Source Software (F/OSS) is a form of software which is developed asynchronously and voluntarily with a group of developers who collaborate over internet. F/OSS has many definitions and different ways of writing (i.e FOSS, OSS or FLOSS – Free Libre Open Source Software), however, main philosophy behind this phenomenon is the freedom to copy, examine and distribute the software produced, therefore contribute to the main body of knowledge.

F/OSS projects mainly start by defining a personal need. A developer sees a gap and necessity behind a requirement and starts building a web page to ease development on the internet with the help of other interested developers. Such projects are initiated by an individual or a small group with an idea for something interesting (Hippel & Krough, 2003). The project initiator (generally the project leader) gives decisions during the lifetime of the project. As more developers, documentation writers, usability experts, bug reports and users enter the project, an ecosystem starts to form around a shared vision. Many F/OSS projects, including Firefox web browser, KDE desktop environment and OpenOffice.org has been successful following all or a part of this progress.

Previous studies (Godfrey & Tu, 2001; Robles, Amor, Gonzalez-Barahona & Herraiz, 2005; Scacchi, 2006) show that open source software projects dominantly grow in either linear or quadratic growth patterns. While many researches rely on small number of projects to discuss the growth of open source projects, Koch's work uses a relatively large sample (4047), indicating that quadratic growth model fits the growth of an individual project better than a linear growth model (Koch, 2007). Similarly, Deshpande and Riehle (2008) have identified 5122 open source projects, written in 30 different programming languages covering 103 different licenses from Ohloh.net by crawling its repository of projects populated since 2005.

Among popular projects, Linux has been given as a successful example to a F/OSS project. Linux operating system (kernel) has been developed since

1991 by Linus Torvalds, who is currently the leader of this project. When the project began, there were only 10.000 lines of code open to the public. After 17 years passed, about 2000 developers are now working on more than 3 million lines of code. While the code written by Linus Torvalds himself constitutes about 2% of the whole kernel, he still takes part in the management lead and decision making part of the project. Contrary to expectations because its development model is not as tightly planned and managed as most industrial software processes, Linux grows at a linear rate for several years (Godfrey & Tu, 2000).

Typically, proprietary software is developed by a closed group, with a manager who directs the programmers towards the goals of the project and company. Programmers are subject to schedules and have a specific list of deliverables (Mockus et. al., 2002). On the contrary, F/OSS projects are developed by volunteers, whose tasks are self-assigned (Crowston et al. 2005). In his essay "the Cathedral and the Bazaar", Raymond (1999) states "given enough eyeballs, all bugs are shallow" in an open source world. In a formal manner, he insists that every problem could be identified and characterized quickly in order to be fixed by a developer, in a healthy environment with beta testers and developers.

Open source applications have been steadily expanding in quality and quantity. A typical Linux distribution may contain more than 2000 applications, utilities and libraries categorized to be used by administrators, software developers, researchers and end users. There have been many successful stories of open source applications and projects covering education, health, telecom and defense sector. Table 2 lists some of the typical application domains and corresponding open and proprietary software applications.

Table 2: List of well known open source applications

Domain	Proprietary	F/OSS counterpart
Browser	Internet Explorer	Firefox
Office automation	MS Office	OpenOffice.org
Web browser	Internet Information Server	Apache
Operating system	Microsoft Windows	Linux, FreeBSD
Instant messaging	MSN Live	Gaim, Kopete
Development editor	Visual Studio	Netbeans, Kdevelop

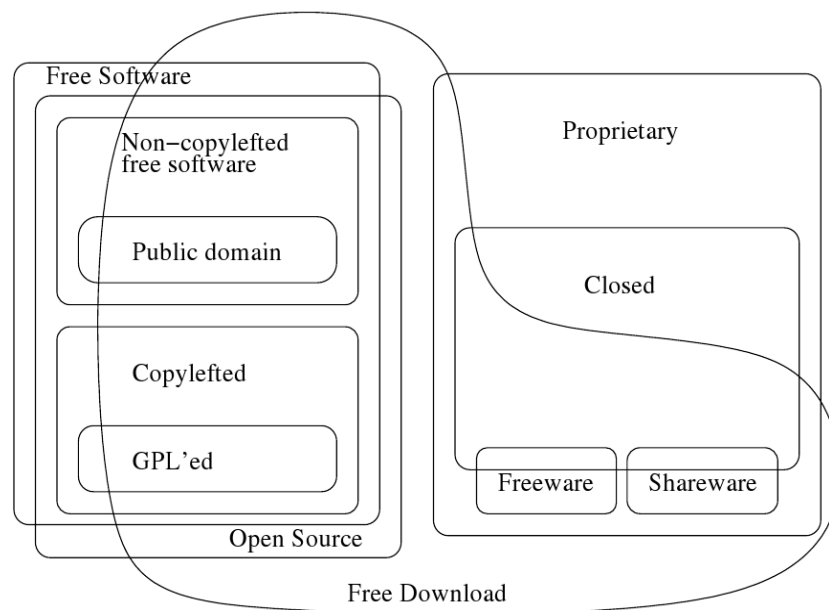
2.5. Open source and proprietary software

In order to understand how F/OSS ecosystem works, it's important to understand the relationship between F/OSS and proprietary software. There are many differences between two, mainly in licensing, distribution and usage. While the key component of an open source software is openness of the source code, proprietary applications are usually distributed as closed source, rendering it very hard if not impossible to analyze the main idea and algorithm behind the product. If a software is free, then it has specific attributes, i.e redistributability, modifiability and freedom to obtain. Free software can come as the form “copylefted software”, in which it's not allowed to force additional restrictions to be added when someone redistributes or remodifies the software. For this reason, each modified copy of the software should also be free. Among all, GNU General Public License is the most prominent form of copylefted free software.

On the other hand, proprietary software has a closed source, usually resulting in restrictions in using, distribution and modification. In the freeware case, the software is distributed unlimited and without a fee. Shareware usually comes with a feature lacking from its original form, trying to force the user to pay for the software. Most of the time, its use is limited to a defined time period. Like freeware, its source code is not exposed. The following

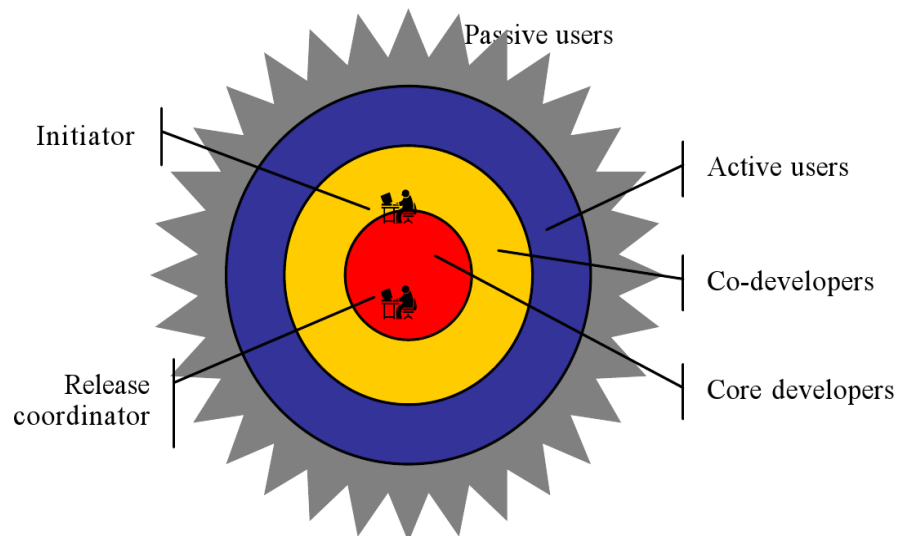
figure shows the relationship between free and proprietary software, in which the freedom of download domain is explicitly defined (Kunei, 2007).

Figure 2: Categories of software



Many authors have defined F/OSS development teams as having an onion-like structure (Cox, 1998; Gacek, 2001; Moon and Sproull, 2000), as shown in Figure 3. According to this figure, at the center are core developers who contribute most of the code and manage the project's future. In a survey, according to Mockus, Fielding and Herbsleb (2002), open source developments has a core of developers who control the code base, and this core is no larger than 10 to 15 people, creating approximately 80% or more of the new functionality. Co-developers surround the onion core. These developers contribute sporadically by reviewing or modifying code or by contributing bug fixes (Crowston, et. al, 2004). Co-developers group are surrounded by users, who can be active or passive. Active users contribute by reporting bugs, writing documentation or translating the user interface.

Figure 3: Onion-like structure of F/OSS developers



2.6. F/OSS and usability

F/OSS contributors consider usability as important. In a survey, it was found that 83% of the questionnaire respondents regarded the importance of usability as either 'high', 'very high' or 'extremely high'. Only 13% considered it 'moderate', 4% stated 'slight' and nobody thought it had no importance. [REF]. One of the developers interviewed explained that some developers see usability as a trivial task which is neither 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" (Andreasen, Nielsen, Schrøder & Stage, 2006). The same survey shows that there is a considerable divergence in the definition of usability among developers. For example, only one out of 24 developers has defined usability within the scope of "satisfaction", and 16 has replied that "efficiency" is related to usability.

Relevant AG conducted a usability test in 2005, studying how two control groups perform predefined tasks using two different operating systems – Linux and Windows XP (Relevant, 2007). One group consisted of 60 users aged 25 to 55 with computer skills but no prior experience with Linux or Windows XP. They tested the preconfigured open-source software

according to various criteria, such as the ease of creating and administrating files, copying CDs and performing some basic office tasks, such as composing a text and sending an e-mail. The other group consisted of 20 users with the same qualifications who performed the exact same tasks on Windows XP. The study findings show that it's almost as easy to perform most major office tasks using Linux as it is using Windows, which has a long history in the consumer market.

Initiatives within EU framework programme (FP) have also taken part to assess the usability of F/OSS products by surveys and tOSSad project² is amongst them. The tests have shown that in average there is a positive opinion among the survey respondents in terms of installation, information presentation, navigation and the overall impression from the operating system (Çetin, Verzulli & Frings, 2006). However, the same project concludes the issues and gaps in the usability area of these products. For example, for some users, one of the disadvantages of Linux operating system is the difficulty of installation of hardware and the learning curve needed in the process of migration from the MS Windows interface. OpenOffice.org also has some issues with insufficient performance compared to MS Office, according to the respondents.

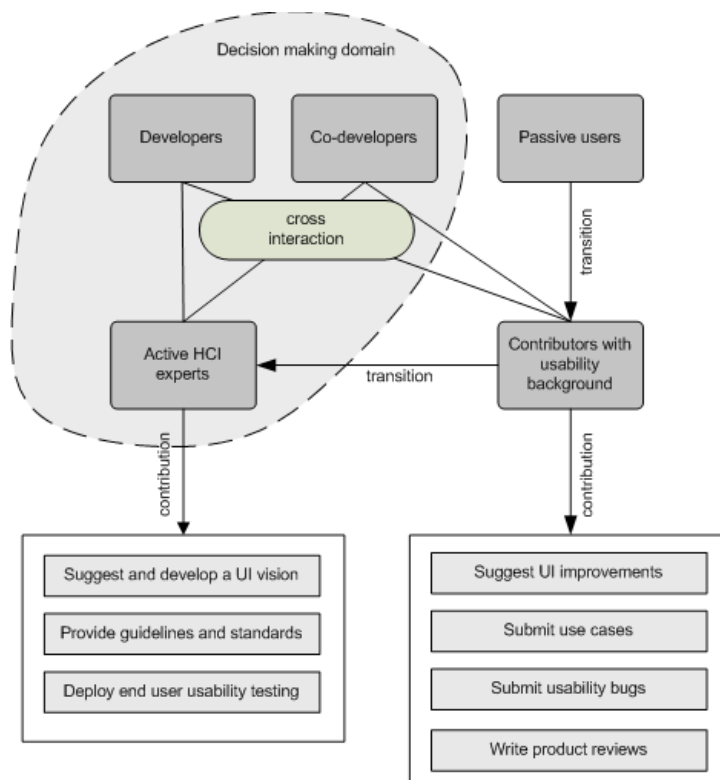
While there are methods to compare the usability of COTS (Commercial off the shelf), or proprietary software with one produced using F/OSS methodologies, licensing issues may prevent benchmarking of the closed source software produced. We are aware of one study which compares Microsoft Excel and OpenOffice.org Calc, a freely distributable and open source spreadsheet application. The conclusion and output of the comparison will not be explained here in detail, however it should be noted that there's a lack of usability comparison methodologies between similar closed and open source software products. On the other hand, there are many factors which effect the outcomes of such comparisons, i.e

² Project web page can be accessed from www.tossad.org

development time, development resources, maturity of the software and prior existence of similar software (Nichols & Twidale, 2002).

While there's little evidence that F/OSS projects are structured in terms of usability knowledge and expertise, there are both active and passive HCI experts, for which we need to identify the amount of contribution for each entity. We also do not have a measured relationship between usability and F/OSS projects. Following figure gives a more detailed overview of a F/OSS project with developers and the process from a passive user to an active HCI expert. Potential tasks are also given in boxes.

Figure 4: Sample tasks of potential usability experts and contributors



While there are very few studies regarding involvement of usability experts in F/OSS projects, many researchers and developers mention the problem of lack of user centric design in this area (Raymond, 1999; Behlendorf, 1999, Frishberg, Dirks, Benson, Nickell & Smith, 2002), hence the weak connection between OSS development approach and usability. Zhao (2007) states that a

community based development model should establish appropriate channels for reporting usability problems. On the other hand, we see that institutionalized usability is gaining momentum in open source projects. It's believed that some studies should be carried inhouse to lower the risk of excessive discussions which do not tie to a result turning into a codebase. OpenSUSE team has undertaken a 3-month usability study with 60 test people compromising of 13 tasks (BetterDesktop, 2007), which was the first in its kind after 15 years. Dan Winship, the UI architect of Novell states (2006):

“[...] If we had proposed the changes on the mailing lists, it would have started a huge discussion about what people hated about the design and how it should be different. And then we could have either (a) completely ignored everyone and done it ourselves anyway, or (b) had a long conversation about the merits of the design and then not actually finished the code in time for NLD10.

Everything good in GNOME is good because one person or a small number of people working closely together made it good. Much of what is bad in GNOME is bad because lots of people have contributed without having a single vision of what the end result is supposed to be.”

According to Winship, endless debates that result in code not actually being written harms the usability process of a product. Instead, designing with a small team is better to handle the usability issues in a software with a unified vision.

After researching the history of F/OSS and usability since 1991, together with the paradigms involved, we can divide the last two decades of F/OSS process into four distinct stages, namely early adopters, emerging era,

modern and paradigm shift. Because of advancements in technology, each era has its own programming languages related to desktop and windowing metaphors. Moreover, we see an enlargement of user base, extending the focus of the operating system environment (both kernel and applications) from system administrators to general purpose users. Table 3 shows an analysis of 20 years of F/OSS desktop (adopted from Çetin & Göktürk, 2007).

Table 3: Analysis of 20 years of F/OSS desktop

	Operating mode	Programming languages	Users (aggregate)	Paradigm
Early adopters (1991-1994)	Mainly CLI, weak interaction, no administrative interfaces	Xlib and motif	System administrators, programmers	Trivial window managers with immature interaction. Low fidelity
Emerging era (1994-1998)	Introduction of WIMP with desktop	Qt, GTK and other high level programming languages	Academics, research institutes	Introduction of desktop metaphor with KDE and GNOME. Switching from keyboard to mouse. Building UI guidelines.
Modern (1998-2003)	New interaction ideas, advanced WIMP technologies	Introduction of platform independent programming languages	Hobbyists, general purpose users	Productivity as advertising image. Usability awareness took off. First articles regarding F/OSS usability. Performance is a consideration. Introduction of paper

				prototyping.
Paradigm shift (2003-2010)	Bloppy UI objects featuring Beryl, Compiz and Compiz Fusion	Abandoning of CLI. More orientation for platform independent languages (i.e Mono, Java, Python). Intuitive GUI design with less or no user	Casual users, users in corporate deployments, introduction of WAF (woman acceptance factor)	Introduction of institutionalized usability. Meeting user needs is a must. Acceptance or rejection stems from usability, not features since many distributions provide same set of tools

The internet is the most striking example of a marketplace, with full of economic opportunity, that has grown out of the open source software community (O'Reilly, 1999). Development of free software on the internet has resulted in a wide range of communication tools and techniques, compared to proprietary software. One could argue that there are distinct points between two approaches in terms of the applications used (i.e bug trackers and forums vs configuration management tools), methodologies deployed (i.e heuristic evaluation vs usability testing laboratories) and communication approaches (i.e random seminars vs regular and planned face to face meetings) which could be a potential future research topic. In a study, Nakakoji, et. al (2002) deploy a wide survey for open source developers, regarding what communication media the developers use in the development of software, claiming there are three distinct open source software project types, namely exploration oriented, utility oriented or service oriented according to its primary goal. This research also finds out that one distinct feature of an OSS project, as compared to the commercial software development, is that members of the OSS project assume certain roles by themselves according to their personal interest in the project, rather than being assigned a task by someone else.

According to Çetin, Verzulli and Frings (2007) there's a lack of research on

how developers really exploit the presence of HCI experts and usability bugs. In their study, KDE (K Desktop Environment) project has been identified as core materials to be investigated, by taking 100 random samples from KDE bug database dated between 2002-2005. Within these samples, usability and non-usability bugs have been identified and classified according to their severity. The bug numbers are randomly taken from bugs.kde.org between years 2001-2005. From the following table, we can immediately see that roughly a quarter of all bugs are usability related.

Table 4: KDE project bugs analysis

	Total number	Avg number of threads	Number of bugs closed
All bugs	100	2,75	80 (%80)
Usability bugs	27	2,78	27 (%100)
Non-usability bugs	73	2,74	53 (%73)

Obviously, there are also issues with open source software development. Many of these problems come from the fact that the software produced is usually a derivative or a collection of other tools and libraries. This causes a NIH (not invented here) syndrome, a form of unwillingness to produce an open source software just because it originates and stems, if not benefits from another open source project.

Healy and Schussman (2003) argue that the importance of hierarchical organization to successful OSS projects is systematically underplayed in the theoretical literature. They suggest that successful OSS projects will tend to have a strong hierarchical component, at least in the ways they manage the relations between lead (and core) developers and other contributors. What they argue is also the weak point of an OSS ecology, especially in circumstances such that many OSS projects cannot cope to progress for a long time, due to lack of resources and interest from a developer point of

view. Daffara (2007) estimates that total number of active open source projects are only at a lower bound of 18.000, compared to over 1.5 million projects listed on Sourceforge site, revealing the short-living duration of F/OSS projects.

Issues stemming from development paths affect the usability process of open source projects. Lack of developers also indirectly mean lack of usability experts. Hierarchical models in which main decisions are taken by lead (core) developers with little deficient user interface design background affect the usability readiness of F/OSS projects, hence the usability of the product. A developer working in a F/OSS project should have explicit rights and titles, together with a strong technical background in order to increase the usability of resulting product. Like Apache project (Apache, 2008), many open source projects are meritocracy, therefore usability experts need to provide a series of strong evidences to support their suggestions, otherwise the developers do not tend to compare them with any other generic contributor with an opinion.

While the survey outputs are elaborated in detail in Chapter 4, it's convenient to list the issues and concerns of communication paths between usability experts and F/OSS developers here, derived from answers to open ended questions.

1. Developers are uninformed about contextual inquiry and user centered requirements process, resulting in a lack of immediate experience of observing product use in the field.
2. Usability experts do not have a detailed understanding of how F/OSS projects work, usually criticizing excessively, rather than contributing, patching to the codebase or solving issues in bug database. While part of this is perception, other part is the attitude of usability experts have the inability or willingness to develop software. In open source world with a limited set of tools and immature development frameworks by means of quality and quantity compared to proprietary

world, a minimum experience of software development mental model is sought to build improved user interfaces.

3. Developers do not have the benefits and outcomes of working with a usability expert. Personal contact between developer and usability expert is often ignored, leaving little or no space to get any form user feedback and to show that the product does actually not work the way the developers thought it would.
4. Lack of scientific research in this field results in a lack of respect in the validity of the field of usability by software developers, and use of scientific usability metrics the developers can relate to, in order to assess their applications' user interface.

In following table, three criteria (employment of HCI experts, communication between HCI experts and project developers and reporting a testing process) is discussed between open source and closed source model, together with advantages and disadvantages. Since some of the arguments are subject to debate, these are enlisted as neutral. In summary, usability in open source model can be stated as weak (employment), virtual (communication) and emerging (reporting and testing). Similarly, usability in closed source model can be described as better than open source model (employment), physical (communication) and well established (reporting and testing).

Table 5: Usability methods and models in distributed and closed groups

Criteria	Model	(Dis)advantages	Closed model	(Dis)advantages
Employment	Weak	(+) fast (+) not subject to a technical barrier (-) low degree	Better than open source model	(+) well defined criteria (-) potentially undiscovered
Communication	Virtual	(-) informal (+/-) online (-) asynchronous (-) usually limited vision (-) control of the software by programmers	Physical	(*) formal (*) well defined (+) direct communication (*) hierarchical, top-down approach
Reporting and testing	Emerging	(-) no well defined tasks and guidelines (-) no well defined templates (+/-) less comprehensive (*) generally indirect communication	Well established	(+) direct communication of experts with users

(+) Advantage	(-) Disadvantage	(*) Neutral
---------------	------------------	-------------

2.7. Software measurement models

Measurement is defined as the process by which numbers or symbols are assigned to attributes of entities in the real world in order to describe them according to clearly defined rules (Fenton, 1994). An entity can be an object (i.e person or software package) or an event. An attribute is the property of the corresponding entity, i.e duration or cost. Generally speaking, it's not possible to improve the products and process of a software project unless a proper measurement and assessment procedure takes place during the lifecycle of the project. In order to have a progress plan, a measurement model should be taken and followed. According to Bicarregui (2000), measure is defined by the following sequential work. i) Identify objects of interest and relationships between them to study. ii) Identify attributes of

formal system to use. iii) Identify relationship between attributes. iv) Check the representation condition.

A proper measurement includes phases of collection, analysis and reporting of software metrics. A metric can be defined as a combination of measures, derived through a mathematical calculation involving several measures. Metrics are used to track progress, determine relative complexity, analyze defects and experimentally validate best practices, and can be identified in two forms: resource metric (i.e effort, training, elapsed time, computer resources, etc) or process metric (i.e use of standards, performance of the running system, method tracking).

With the help of metrics, software development teams assess their applications and build effective and efficient software, providing measures that management level staff can understand and analyze issues and gaps. To provide practical value, metrics should identify frequency of measurement, frequency of review, source of data, rationale for introducing the measure, who will act on the data, and the purpose of the measure (Neely 1998). For scientific, quantitative rigor, metrics should exhibit, at a minimum, construct validity and reliability (Cook and Campbell 1979).

Assignment of simple numbers to hypothesis without knowing the empirical evidence is a major mistake (Zuse, 2008). Therefore, a theory building process should take place in order to watch a valid progress. In order to provide empirical results, numbers or symbols are first gathered from the project. These numbers are then processed in order to derive a set of metrics. The final stage of measurement is interpreting the numbers gathered. Likewise, according to Nielsen (1996), it is somewhat simplistic to assume that usability can be measured by a single number and there is no such thing as a "user friendly" product.

There are ways of which software metrics are collected and interpreted. Examples include order of growth, source lines of code, function point analysis, bugs per line of code, number of lines of customer requirements

and number of classes and interfaces. On the other hand, metrics can be stated as unethical, demeaning, skewing, inaccurate, suboptimal and uneconomical, therefore it poses potential weaknesses and receives criticisms (Wikipedia, 2000).

2.8. Measurement frameworks

Empirical techniques are divided into informally observing, formal experiments, industrial case studies and benchmarking exercises or surveys. “There is no objective way of ensuring that one program is equivalent to another, so that it is impossible to ensure that experimental object (e. g. designs, programs, modules etc.) are directly comparable. However, the characteristics of a specific experimental object may influence the experimental results (Kitchenham, 1997). Measurement must be driven by goals and models for the processes, products and projects (Basili, 2002).

Measurement models basically fall in two categories, namely top-down and bottom-up (Pfleeger & Rombach, 1994). Bottom-up approaches start with measurable observations and build management objectives and goals, whereas top-down approaches help to derive useful measures from goals and interpret the collected data in the context of the goals of interest (Wangenheim, Punter and Anacleto, 2003). According to Rombach (1991), top-down approaches have shown to support the adequacy, consistency and completeness of the measurement plan and to help manage the complexity of measurement programs. Zelkowitz (1998) observes that more than 20% of the papers in the journal IEEE Transactions on Software Engineering have no validation component (either experimental or theoretical). This percentage dropped from 36 percent in 1985 to 29 percent in 1990 and only 19 percent in 1995.

Quantitative Measurement of Program Quality

Rubey and Hartwick (1968) propose a set of quality attributes of software used in spaceborne software, deriving 7 high level attributes, each of which are categorized into 6 to 12 subattributes. According to Rubey & Hartwick,

this model can be used for measurement or comparison of program quality, as well as for evaluating the importance of various aspects of the programming environment and directing the development of program. The model defines the following attributes, of which not all of them are targeted to a specific programming language.

A1: Mathematical calculations are corrected performed.

A2: The program is logically correct.

A3: There is no interference between program entities.

A4: Computation time and memory usage are optimized.

A5: The program is intelligible.

A6: The program is easy to modify.

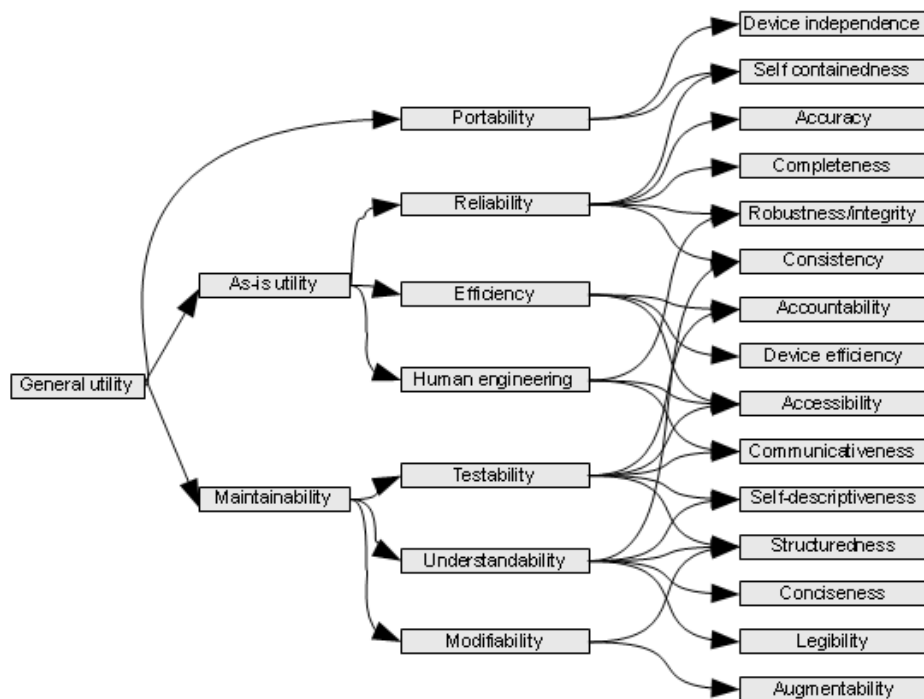
A7: The program is easy to learn and use.

In this method, a sub-attribute can be placed in multiple attributes, and there's a complex interrelationship between sub-attributes.

Software Quality Characteristics (SQC)

In a study, Boehm, Brown, and Lipow (1976) establish a conceptual framework and some key initial results in the analysis of the characteristics of software quality. Derived from a previous study (Boehm et al, 1973) they have developed and defined an initial set of software characteristics, namely understandability, completeness, conciseness, portability, consistency, maintainability, testability, usability, reliability, structuredness and efficiency. Their work is cited as one of the first researches to provide a well defined framework for assessing the issues associated with software quality.

Figure 5: Software Quality Characteristics (Boehm, Brown & Lipow, 1976)



Software Quality Metrics (SQM)

After Boehm, Murine (1968) has developed “Software Quality Metrics” (SQM) which defines 12 factors, 23 criteria and 400 metrics, by adopting portions of Boehm's research. SQM approach can be stated as “top-down”, since Murine states that data gathered for the metrics must be interpreted in the light of criteria and factors. SQM is the one of the most used metrics in literature (Carpenter et. al, 1983).

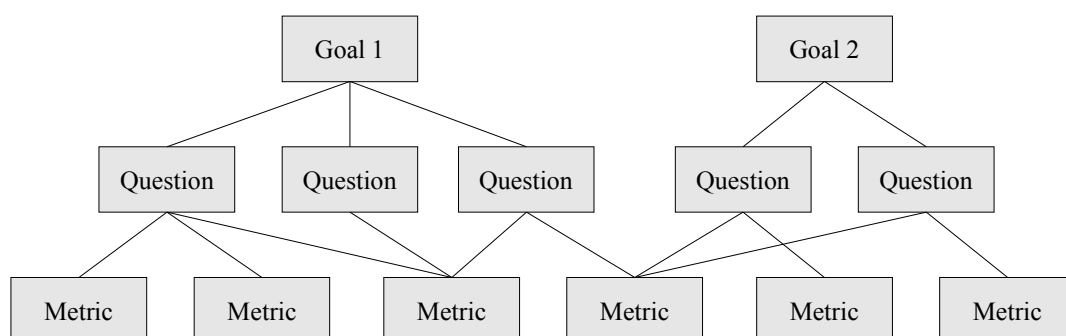
Goal-Question-Metric (GQM)

GQM paradigm is a widely used measurement method which constitutes a framework to develop a metrics program. Initially developed by Basili and Weiss (1981), GQM provides a mechanism to identify, plan, build and analyze the measurement structure. GQM begins with generating a set of goals, continuing with deriving and developing a set of metrics. Within this framework, Basili, Caldiera and Rombach (1994) define goal, question and metric as follows:

- **Goal (conceptual level):** Defined for an object for a variety of reasons, with respect to various models of quality, from various points of view, relative to a particular environment. Objects of measurement are products, processes and resources.
- **Question (operational level):** Comprised of a set of questions to characterize the object of measurement (product, process, resource) with respect to a selected quality issue and to determine its quality from the selected viewpoint.
- **Metric (quantitative level):** A set of data is associated with every question to answer in a quantitative way. Data can be objective, depending only on the object that is being measured and not the viewpoint from which they are taken, or subjective, depending on both the object that is measured and the viewpoint from which they are taken.

Figure below depicts the GQM model hierarchy, showing the main parts and their relationships between (adopted from Basili, Caldiera and Rombach, 1994).

Figure 6: GQM model hierarchical structure



Unlike SQC and QFD, which are limited to product quality, GQM can also be used for process quality.

Quality Function Deployment

Quality Function Deployment (QFD) is “an overall concept that provides a

means of translating customer requirements into the appropriate technical requirements for each stage of product development and production (i.e., product design and engineering, prototype evaluation, production)” (Sullivan, 1986). In the early times, QFD applications focused on software, i.e decision support systems (Sarkis and Liles, 1995), human–machine interface (Nibbelke et al., 2001), information systems (Chang and Lin, 1991) and web pages (Tan et al., 1998).

2.9. Current usability measurement models

Usability is relative, that is, it's meaningful to speak about a set of tasks which a set of users can achieve. Therefore usability is usually measured in a relative scale and not absolute, where one product context is assessed in its own domain. Researchers have adopted many methods and procedures of testing usability and specifying usability measures. Kirakowski (1994) defines two approaches towards measuring user satisfaction and quality of use. One approach is assessing what can be considered good for an interface, and then evaluating the extent to which a system exhibits these properties. The other approach measures the reactions of end users with which the system is evaluated. The first approach mainly depends on the evaluator and not the end user, ignoring the necessity of their opinion. The latter only evaluates the user interface and does not consider the ideas and suggestions of the experts.

There are many definitions on the metrics and evaluation methods of usability and user centered design. ISO 9241-13 defines the dimensions and definition of usability (ISO, 1998), which is by far one of the most referenced definitions by academia. Sauro and Kindlund have turned the ANSI and ISO definitions of usability into a single, standardized and summated usability metric which provides one continuous variable for usability evaluations that can be used for regression analysis and usability reporting (Sauro & Kindlund, 2005). This research has its approach to simplify all aspects of usability into a single score by using 4 data sets totaling 1860 task

observations which in turn shows that these aspects of usability are correlated, equally weighted, presenting a quantitative model for usability.

Brooke (1996) also describes a system usability scale (SUS) which can be used for global assessment of system usability. He mentions the need for a method to allow low cost assessments of usability in industrial systems evaluation. As well as defining how to implement a global assessment of usability, Brooke also describes a questionnaire method using a pool of 50 potential items using a Likert scale. In another study, Lucey (1991) demonstrates that this short ten-item scale has a reliability of 0.85. According to Kirakowski, "Lucey's figure may be an underestimate because she reports data from only 77 users".

Other questionnaire sets for measuring perceived satisfaction of end users include QUIS (Shneiderman, 1987), SUMI (Kirakowski & Corbett, 1993) and QUIS (Chin, Diehl & Norman, 1988). Among these, SUMI is defined as "*an industry standard evaluation questionnaire for assessing quality of use of software by end users*" by Kirakowski (2007). SUMI, which consists of a 50-item user satisfaction questionnaire measuring specific usability areas such as effectiveness, efficiency, control, helpfulness and learnability, was developed as a part of MUSiC (Metrics for Usability Standards in Computing model) project by Bevan and Macleod (1995, 1997). MUSiC consisted of a measurement framework which was used to assess task effectiveness, temporal efficiency and length of productive period.

Heuristic evaluation (Nielsen, 1993a) is one of the most popular methods of usability inspection, carried by a group of experts in order to find usability problems in the design in order to evaluate its compliance with well known usability principles. It's one of the basic approaches Kirakowski (1994) defines, having the same set of weaknesses, i.e lack of user involvement. Nielsen defines ten heuristic methods (1994), and apart from this, Muller adds 4 more heuristics to this set (Muller et al, 1995) and then defines e-commerce specific heuristic set including 13 heuristics. Each year, heuristic

sets become more and more complex and well defined in order to provide the most accurate and valid output. This effort has proven to be quite successful, as Jeffries, et al (1991) found out that “heuristic evaluation by several UI specialists found the most serious problems with the least amount of effort” among four evaluation techniques including heuristic evaluation, software guidelines, cognitive walkthrough and usability testing.

Traditional usability approaches focus on end products. Nielsen (2003) discusses a method to measure 4 different web based systems (home banking, cash register, security, hypertext) and shows how usability metrics are measured and further refined into estimates of improvements in overall usability. He adds (2002) “Among other things, metrics can help managers track design progress and support decisions about when to release a product. In his example, Nielsen uses geometric mean in order to compute the improvement of the tasks with which a single number represents the time it takes for users to finish with their tasks. From these examples we can deduce that measurable variables are gathered from heuristic evaluation, end user (usability) testing and cognitive walkthrough.

In their paper Seffah, et. al (2006) propose a consolidated model for usability measurement called QUIM (Quality in Use Integrated Measurement), which is a hierarchical model decomposing usability into factors, criteria and specific metrics. Authors claim that QUIM follows the IEEE 1061 standard (Software Quality Metrics Methodology), which outlines methods for establishing quality requirements as well as identifying, implementing, analyzing, and validating both process and product quality metrics. According to Seffah et. al, “it may be possible to create a knowledge base that can be used for usability prediction, that is, as an automated quality assessment tool that reduces design, testing, and maintenance time”. QUIM is not a revolutionary model, rather it combines models previously defined in their paper. Based on their review of the usability measurement literature, 10 usability factors briefly included in the QUIM consolidated model are efficiency, effectiveness, productivity, satisfaction, learnability, safety,

trustfulness, accessibility, universality and usefulness. Moreover, 126 metrics have been defined which are either countable (which are collected from raw data) or uncountable.

2.10. Conclusion

Open source projects are developed online and asynchronously where developers work on a set of goals and objectives in order to produce a software result using project resources and applications. Developers mainly do not see or even hear each other, trying to communicate in a proper media such as mailing lists and forums. Therefore, quality of the project attributes and usability factors directly affect the overall product usability and a new measurement method and project usability metrics is required. Defining the metrics and the corresponding measurement model which effect usability factor of the project is also needed. Once the metrics are extracted from project patterns, we need define and assess these metrics in order to develop a model.

This chapter discusses the current trend and state of the art in 3 domains: usability, open source and software measurement models. While the current level of technology is relatively high in each domain, there's a lack of interaction among them. Although there have been a number of improvements in usability of F/OSS projects for the last 5 years, there are a number of unsolved and critical problems that prevent F/OSS developers and HCI experts to build a usability centric framework on top of their projects.

3. Methodology

3.1. Introduction

The inclination of developers towards usability determines the process and project. In order to create a competitive advantage and growth, project developers constantly need to seek a “usability culture” within their distributed project. Unfortunately, there are lots of references pointing to how to define the usability of a product, but there is little known about how to define and assess the usability of a project which produces a product. Our contribution is to define a set of project usability metrics which can, in turn be discussed in parallel with software usability, where interface improvements and quality are discussed via heuristics, well defined tests and tasks.

This chapter will justify the methodology used to build a measurement model and its corresponding metrics. In order to validate the metric table, a survey has been completed. The unit of analysis, test subjects, sampling frame, sample size for the survey and procedures to handle the survey data are also clarified in this section.

3.2. Justification for the paradigm

Our main research question in this field is, can we define a single set of metrics which can be used to measure “usability maturity” and “usability readiness” of a distributed project and how can we define and justify our set of checklists by using qualitative and quantitative data? While the first part is straightforward, the second part (justification) is hard due to the fact that calculating the usability readiness of a project requires subjective evaluation.

Here, we give a basic metric definition and justification of usability metrics of distributed projects, focusing on those developing free and open source software. This research contributes to measuring the degree of unindustrialized usability among distributed projects with the help of authors' own experiences in managing F/OSS projects. A matrix which will include a

metrics analysis of a distributed project with criteria defined by literature review and survey analysis will be created. In this matrix, each criteria (i.e project web site, tasks and guidelines, user centered design and awareness) will be mapped to a longer definition and a range.

The outputs of this metric model can be used to produce a regression analysis and prediction (i.e including forecasting of time series data). As a future work, more case studies could be presented to provide evaluative data to study the relationship among usability metrics of different F/OSS projects. In this sense, research including regression analysis could be considered to analytically compare the collected data.

In order to fulfill the goal to provide a better understanding and assessment of usability metrics in open source projects, a new usability metric model is designed for quantitative Usability Measurement of Open Source projects (abbreviated as UMOS). These set of metrics differ from others in the sense that UMOS focuses on the project progress and process rather than quantitative analysis of the source code of the corresponding project. Therefore, the offered metric presents no interaction with source code analysis.

It's important to develop a set of metrics for project usability for the following reasons. First, we will be able to have a predefined set of metrics to assess a defined group of projects. Second, having such a metric will involve project developers to assess their projects in terms of usability metrics. While these two reasons are interchangeable, it's important to understand for developers to have an idea how their project performs regarding F/OSS usability readiness.

In order to provide a model hindering the complexity and providing the simplicity in itself, it should be guaranteed that following criteria are fulfilled:

- Project usability metrics model should be easy to use and understand
- The metrics model should cover a wide range of projects, taking into

consideration the different states and circumstances of projects.

- It should provide enough information to support the qualitative & quantitative decision making and support for increasing project awareness.
- It should support for increasing project quality.
- It should enable to make predictions and assessments.

3.3. Research procedure

The basic steps taken in this research have been depicted in Figure 1. According to this figure, the main parts of the research procedure is as follows:

1. *Developing and understanding of application domain, by conducting exploratory research and deriving a research problem.* Here, useful variables can be derived enough to formulate a data set. A focus on the problem domain and supporting areas has been given in order to demonstrate a valid research problem and generate valid results.
2. *Identifying the sample population, and surveying this domain.* This process also includes cleaning and reduction of data and removing doubles.
3. *Interpretation of results.* This is done with the help of analyzing previous models
4. *Recommending a metric model* by identifying and setting indicators, together with providing corresponding descriptions and scales.

In this thesis, 3 phases to assess project usability have been defined as follows.

1. Assess vital project elements and indicators (i.e number of total usability experts, ratio of usability experts to developers, heuristic evaluations defined or not, guidelines defined or not, bug tracker distinctly accepts usability bugs or not, frequency of usability tests,

etc).

2. Define a metric set for each item based on organizational requirement and developer priorities.
3. Produce a matrix and calculate the overall score.

Indicator categories are defined below, based on our review on usability measurement literature.

1. **User centered design:** Gives insight about how the project stakeholders see usability and user centered design. It includes how usability is mentioned in project aims, whether a set of heuristic evaluations and usability tests are defined in project tasks or not.
2. **Developer base:** Draws a picture of how well the project developers are suited to a user centered project. It includes metrics of user profile definition, number of graphics designers and HCI experts, ratio of graphics designers and usability experts to total number of developers having access to project repository.
3. **Bug tracking:** Introduces a means of how bug reporting tool in use is exploited to support developers and track design progress and support decisions. It includes whether usability related bugs can be prioritized and handled in a different category of its own.
4. **Project web site:** Shows different aspects of project site and defines whether the project is successful or not by means of introducing an easy to use and explore content navigation.
5. **Tasks and guidelines:** Gives an overview of if and how tasks and guidelines (i.e user interface guidelines, usability test templates and tutorials, guides for icon design etc) are provided for developers.
6. **Accessibility, internationalization and localization:** Informs how project benefits from an accessibility framework for disabled users, also uses ways to assist translators to submit their work to project

repository.

The factors above are mostly independent from each other, however a thorough investigation reveals that there are few relationships between some of the items. For example, “accessibility management” and “i18n and localization management” can either be located in “accessibility, internationalization and localization” or “project web site”. On the other hand, some of the terms which are found in other model criteria (i.e usability, task efficiency etc) is not mentioned in this model.

Indicators given above have been identified by investigating the survey given to 36 HCI experts and 42 F/OSS project developers. Using our review of existing usability measurement models together with the developer and HCI expert survey, we identified a total of 6 major criteria and 19 specific usability metrics, depicted in Table 12 in Section 4.4. Some metrics are derived using other metrics, and some are directly countable. Examples of countable metrics include number of HCI experts and number of graphics designers. Refined (calculable) metrics are calculated using formulas or generating raw observational data by heuristics. The aforementioned table provides a list of metric criteria and their corresponding definition with chart samples. Type of scales in the table can be one of the following.

- **Nominal scale (classification):** This is a list of categories to which objects can be classified. Within this context, only answers like “Yes” or “No” are counted as a nominal scale.
- **Absolute scale:** Only used for counting (i.e number of developers in a project).
- **Ordinal scale:** No size, no comparison of differences (i.e Likert scale). A Likert scale is used while assigning value to an object based on ranking with respect to one another.

Interval scale for quantifying differences (i.e date) and ratio scale (i.e length) are inappropriate for giving a value for each criteria, and are not used within

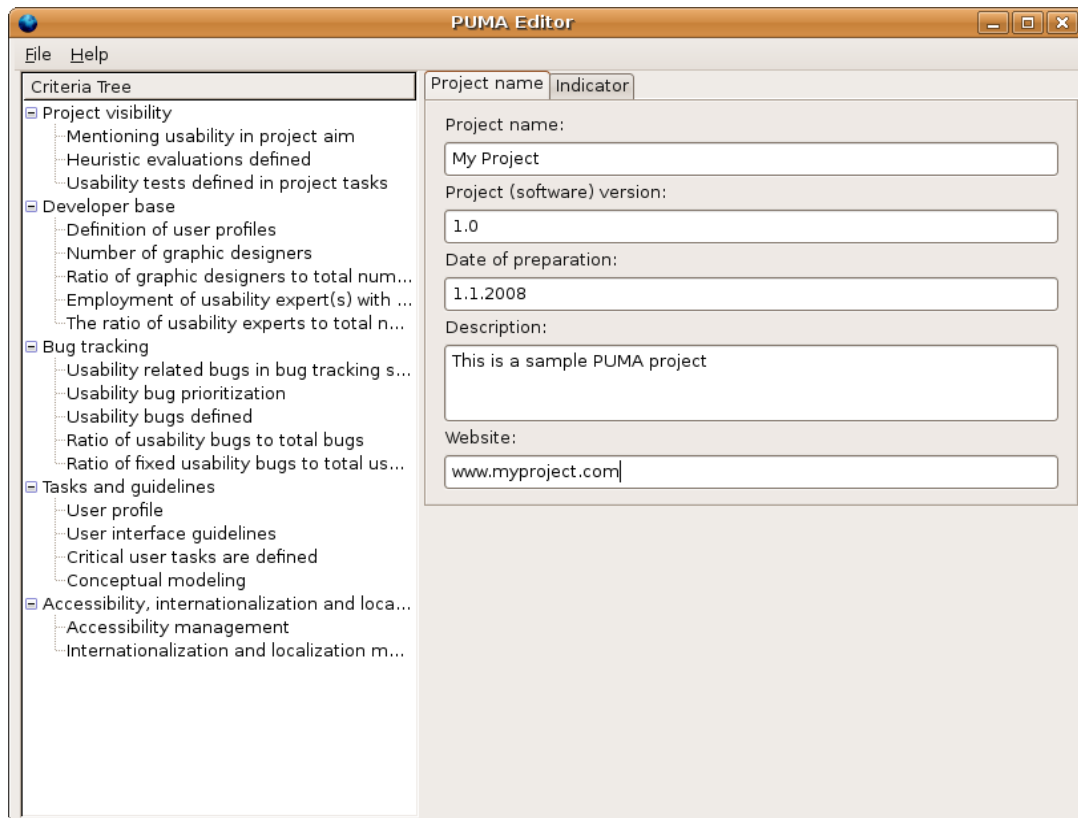
the metric system.

3.4. Project usability metrics aggregator

In order to easily enter the data of criteria for a specific project and calculate the overall score of the usability matrix, the author has developed an editor in order to streamline the input of project stakeholders. This editor can be used for those who are not experienced with data entry process and half-complex mathematical foundations behind the framework. The main interface of the Project Usability Metrics Aggregator (PUMA) is depicted in Figure 7. This editor is used to assist F/OSS project coordinator to insert and browse the usability metrics defined in section 3. Moreover, a search facility is developed to locate a particular term. PUMA also defines each metric succinctly to give a broad overview of what is meant by the criteria. While the software uses the metric model developed in this thesis, a user can modify the metrics and derive another model satisfying the requirements of another online project.

After the software is run, the user is presented with two tables. On the left, each category is defined together with their indicators. On the right, each indicator is elaborated, together with an input mechanism for each scale type. User can fill in the answers, saving the results in XML format. Once the new measurement data is entered, PUMA saves it for later modification or generate a graphical analysis in HTML format which contains the metrics entered together with their interrelations and calculations.

Figure 7: Main screenshot of editor showing the introduction page



PUMA has the following features.

1. **Multi-platform:** PUMA can be run from any browser via any operating system due to platform-independence of the software and utilities used. XUL (XML User Library) is used during the development of PUMA, connecting each UI item with a Javascript function to read or write an XML file.
2. **Multi-view:** It's possible to see all the criteria on the left hand side, by choosing a criteria and entering a value and/or a comment for each indicator.
3. **Extensibility:** Each end user can modify the template XML file by adding or removing criteria to suit their needs. Therefore, PUMA could be used for measurement purposes in other domains.
4. **Help and documentation:** PUMA has an internal help to ease end

users get specific information about the usage of the software.

PUMA XML file has a distinctive suffix of .puma, and has the following format. Bold font type indicates the input coming from the user, and italic font type shows that given phrase is read from PUMA XML template.

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <header>
    <projectname>Name of the project</projectname>
    <release>1.0</release>
    <url>URL of project web site (format: http://www.url.com)</url>
    <date>Date of preparation</date>
    <desc>Short description of the project</desc>
  </header>
  <section name="category name" title="category title">
    <desc>Category description</desc>
    <element name="IndicatorID" title="Indicator title">
      <desc>Indicator description</desc>
      <value>Value</value>
      <comments>Comments (if any)</comments>
    </element>
  </section>
</document>
```

3.5. Survey planning

In order to examine how usability experts and developers work, two surveys have been generated and deployed – one for usability experts, and the other for F/OSS developers. The main idea behind two surveys is to correlate the survey questions according to the type of applicants. Since there is no published evidence that the concurrent fielding of a survey via a conventional mode and an Internet-based mode results in any significant improvement in response rates (Fricker and Schonlau, 2002). To improve coverage, Dillman (2000) recommends a mixed-mode strategy for contact, using both e-mail

and postal mail for prenotification. Similarly, using mixed response modes, such as the web and e-mail, would increase coverage of F/OSS developers and usability experts' coverage.

3.6. Aim and motivation of the survey

Within the survey, main answers to these questions has been sought:

1. What are the main characteristics of a project that encompasses a user centered design (UCD) methodology?
2. What kind of media, tools and frameworks do F/OSS developers and usability experts use in common to discuss and evaluate usability issues?
3. What distinguishes F/OSS projects that are especially innovative in their functionality and interface design?

There's no clear definition or national/global data which holds the developers demographic information, the only and obvious way of surveying is that survey respondents are drawn randomly from general population. In order to receive more factual data and sake of completeness, a number of survey respondents has been chosen for a specific survey, nevertheless they might not be representative of the general population for certain criteria. While the main aim is to reach as many respondents as possible, It's not possible to estimate how many people has read the invitation e-mail.

The following e-mail was used to send the invitation to public mailing lists and forums as well as private people. At the end of the e-mail, there's a discrete paragraph showing that the anonymity of the respondents will be protected, since the survey does not consider the private information of the respondents (including gender, country and the name of the project). Moreover, there are some questions, for which the inclusion of private data could result in a negative approach for that specific project the user belongs to.

The survey web page started with an informed consent form for survey

respondents including the title of the survey, aims and contact information. As the survey finishes, the respondents are asked to contact to author in case they want the raw results.

We are inviting you to participate in a doctorate study that investigates how developers and HCI experts work in a collaborative project. You will be asked to answer a set of questions regarding F/OSS projects you take part in. Your participation in this survey will provide information that could be used to improve how HCI experts and developers interact in a F/OSS project.

Online survey takes only 15 minutes to complete and will allow us to have an understanding of interaction patterns among project participants.

If you are a developer, click on:

http://www.surveymonkey.com/s.aspx?sm=Vvf6V9qqg5edTCYFe_2fZJng_3d_3d

If you are usability expert, click on:

http://www.surveymonkey.com/s.aspx?sm=_2fBlhjV3nPhLhe_2fFQyGoJYA_3d_3d

in order to complete this short survey. Please circulate this email to your friends and contacts so that we can have the widest possible feedback.

Note: demographic inputs of this study will be kept strictly confidential, i.e your name and any other private data will not appear anywhere publicly.

The target respondents of the survey were twofold:

1. **Usability experts** who are assumed to be working in a F/OSS project. Their ideas stemming from long term experiences devoted to increasing the usability awareness of F/OSS projects are used to derive the measurement metrics table in Chapter 4.
2. **Developers** who take part in a F/OSS project by committing code, implementing a feature, sending a bug report or reviewing a part of the software. F/OSS developers are secondary targets having a crucial role in developing usable applications, but often fail to do so.

3.7. Structure of the survey

We conducted the survey by means of a web based questionnaire system hosted on www.surveymonkey.com website. Participants were not forced to provide answers before going to the next page, and with the help of the commercial survey framework, each page was protected from being cached. While the questionnaire was open to anyone, the author has sent invitation e-mails to open mailing lists specifically used by HCI experts and F/OSS developers in order to set the focus on two kinds of respondents. Invitation e-mail included a request to forward the e-mail to other interested parties. The author has sent to a total of 12 mailing addresses and 76 individuals whom are known to be competent and experienced in usability and open source development domains.

The questionnaire included 26 and 35 questions for usability experts and open source developers, respectively. Some of the questions were duplicates in order to cross compare the responses of two groups. Annex 1 includes a list of survey questions.

3.8. Error reduction in survey

In our case, survey respondents are HCI experts who work in F/OSS domain, and F/OSS developers who have a project to work on. Both population has a sample group available online without conducting direct face to face or phone surveys. Therefore, a survey methodology comprising of a special application via web is chosen, eliminating the burden to try to contact with every potential survey respondents. Heberlein and Baumgartner (1978) used a technique of meta analysis to test the predictability of 71 characteristics on response rate. Their findings related to positive and negative effects have been examined and applied to this survey, mentioned in the first column following table along with other references affecting the success of survey. The second column of the table is used for revealing what has been done in order to increase the response rate in survey.

Table 6: Error reduction methods used in user survey

Results of Heberlein and Baumgartner	Application of results in the survey
<u>The number of contacts:</u> More contacts will increase the response rate. Advance letters, postcards, follow-ups that include additional copies of questionnaires, and even telephone calls are all examples of such contacts.	In order to increase the response rate and number of contacts, an advance letter has been sent to academics and domain experts previously defined.
<u>Employee population:</u> Samples from some special subgroups, such as employees from certain occupations, are more likely to return survey research than the general population.	Survey requests have been sent to small groups, in which many list members know each other for quite a long time.
<u>Special third contact:</u> Following up the advance letter and initial follow-up with the use of special mailing procedures, such as certified mail or special delivery, or with personal or telephone contact increases the response rate.	A special third contact mechanism hasn't been deployed.
<u>Plain questionnaire:</u> A plain questionnaire obtains higher response rate than a fancy version of that questionnaire.	Survey benefited from utilization of simple page layout design features in order to benefit from higher completion rates or better quality data
Negative effect on response rate	
<u>Marketing research sponsorship:</u> Marketing research surveys in which the information will benefit the firm have lower response rates.	Survey was not sponsored by any company, including a market research organization.
<u>General population:</u> Samples drawn from the general population have lower response rates.	In order to increase response rate, 10 open source mailing list has been selected and an e-mail is sent to mailing list participants to draw the samples from a dedicated set of people.
<u>Questionnaire length:</u> Longer questionnaires produce lower response rates (Heberlein and Baumgartner, 1978).	Survey includes less than 30 questions, with most of them requiring a single answer from a list of choices. It has been estimated that the whole survey, including the open ended questions, take no more than 15 minutes to complete.

The following compilation method is used in order to gather and revise the raw data: i) Unmodified data is extracted from survey site, and ii) This data has been cleared down. Duplicate data entries have been removed in order to provide a clean data for analysis. No reminders have been posted to the individuals and mailing lists.

4. Analysis of data

4.1. Patterns of data

This section gives the main findings derived from survey responses. Compilations from survey and concentrated findings will be used to derive the measurement method, particularly the table in section X.Y. Analyzing the survey which was answered by 36 HCI experts and 42 developers has yielded following results. The first part (chapter 4.1) includes the itemized set of data, whereas chapter 4.2 deducts information from the first part. Each outcome is defined by a latter group in order to reference throughout the subchapters in an easy manner.

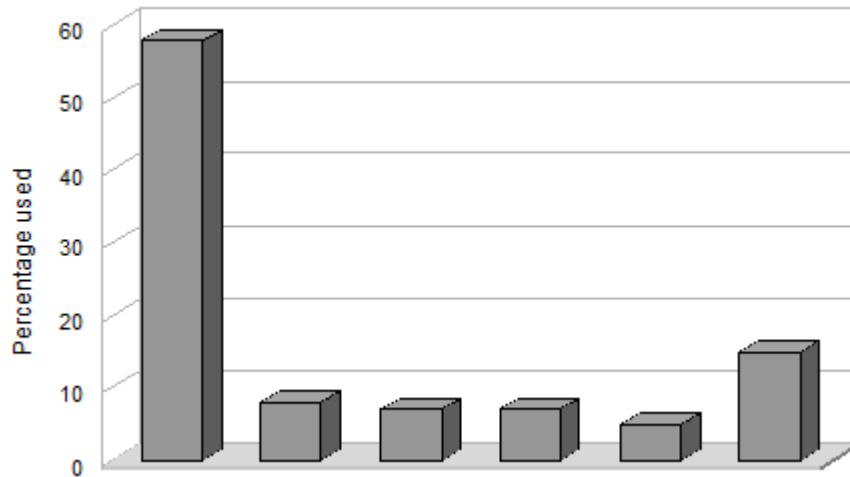
75% of the respondents work as a usability expert in at least one F/OSS project (*only asked to usability experts*) **[A1]**. Among them, 70.3% (19 out of 27) is paid by a company and/or organization because they work as a usability expert or UI designer **[A2]**. 55.5% of the respondents say “*There’s a user centered design process in my project or organization of which we follow*” **[A3]**. This shows that almost half of the usability experts receive an income for their jobs in open source projects.

Table 7: Developers according to their job status

		Paid	Unpaid
		70.3%	29.7%
Works as an expert	75%		
Does not work	25%	N/A	

Among usability experts and developers, more than half (66.1% - 43 out of 65) use Bugzilla to report bugs. 16.9% does not use any reporting tool. Other tools used for tracking bugs include Trackback, Jira, Trac, Gforge, Scarab, Berlios and Launchpad. Bugzilla is the clear winner among all bug reporting tools **[A4]**.

Figure 8: Percentage of bug trackers used



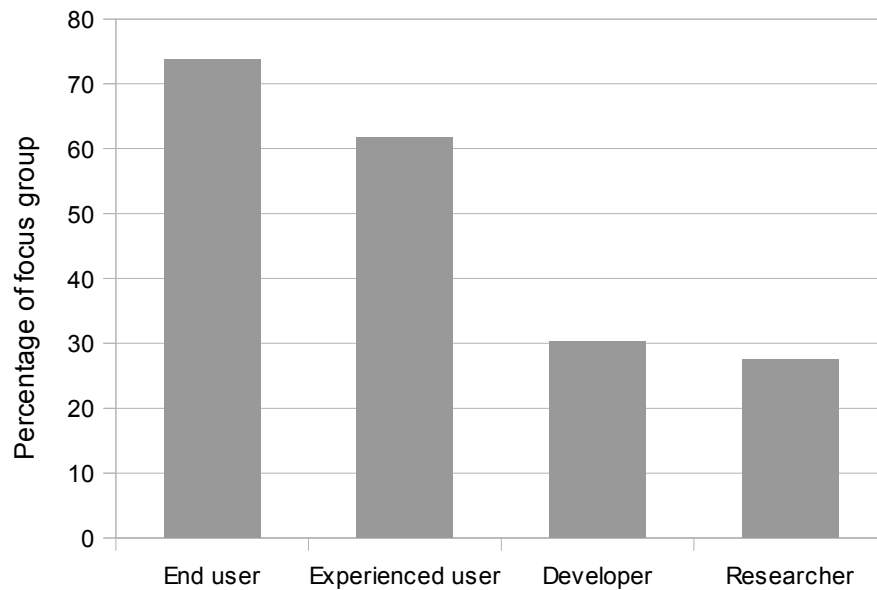
For discussing usability issues, 92.5% of the respondents use at least one type of media (i.e mailing lists, IRC, instant messenger or bug tracker). Among them, mailing list and IRC are the most popular applications (64.1% and 17.9% respectively) and instant messenger is the least popular (7.4%) [A5]. This question is asked to both group. While 58.6% of the respondents explicitly mention the bug tracker they use is an ineffective tool for usability reporting, 14.2% say otherwise. There is no relationship between the effectiveness and the name of the reporting tool. There's no big difference observed between the answers of usability experts and developers [A6]. 28.3% of the projects have a specific component dedicated to usability inside the bug tracker [A9].

Table 8: Use of media for usability discussions

A media is used to discuss usability issues	92.5%	Mailing lists	IRC	Bug tracker	Instant messenger
		64.1%	17.9%	10.6%	7.4%
A media is not used to discuss usability issues	7.5%	N/A			

“End user” is the biggest focus group, with 73.7%. Experienced user is another big chunk of focus group with 61.8%. Developer and researcher are 30.2% and 27.6% respectively (*asked to both group*) [A7] .

Figure 9: Percentage of focus group



There's a wide selection of the usability evaluation methods used in F/OSS projects. Expert review, which relies on feedback of experts has been used in 69.8% of the projects. Then comes interface guidelines (57.5%), user testing (52.05%) and getting feedback from early adopters (42.4%) . Focus groups is the least preferred method chosen in (21.9%) of the projects. Paper prototyping, heuristic evaluation and expert review are 3 main usability evaluation methods in which usability experts rely on more than developers [A8]. The following table shows the difference in preference of the aforementioned methods. This question is asked to both group.

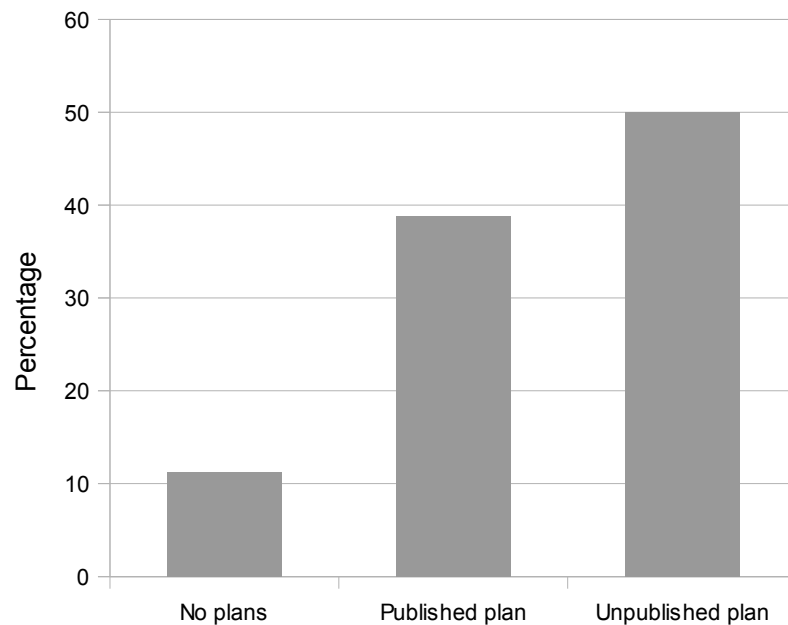
Table 9: Choice of evaluation method. Source: Analysis of survey data.

Choice of evaluation method	Survey respondents	
	Usability experts	Developers
Paper prototyping	44.4% (16)	22.2% (8)
Heuristic evaluation	50.0% (18)	27% (11)
Expert review	77.7% (28)	57.5% (23)

There's a balance between projects which have experience in deploying surveys asking users' opinions regarding application usability (52.2%) and which have not (47.8%) (*asked to both group*) [A10]. 40% of the usability experts know of people who cannot submit bugs because of the usability problems found in the bug tracker (*asked to usability experts*). Likewise, 48.2% of the developers have received at least one complaint from a user who cannot submit bugs because of the usability problems found in the bug tracker (*asked to developers*) [A11].

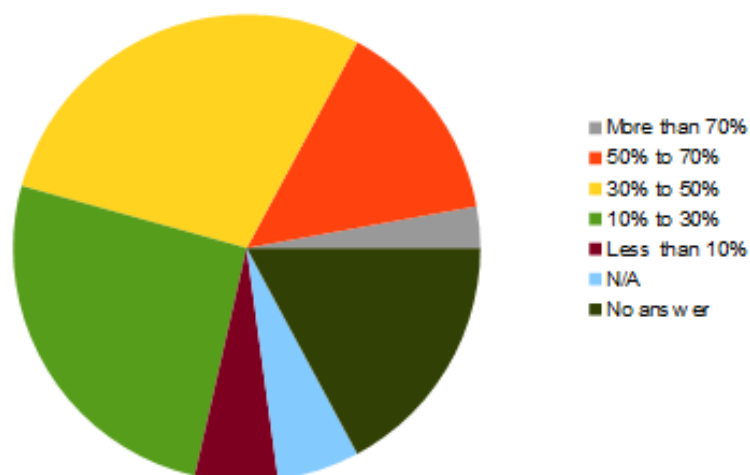
11.2% of the projects do not have any plans for gathering usability requirements, building user interface guidelines or refactoring and/or amending product user interface developed within the project. 50% of the usability experts mention that there's a plan, however it's not published. Remaining 38.8% has a formed plan, which is published in project web page (*asked to usability experts*) [A15].

Figure 10: Projects having plans for usability methodologies



Keeping weekly project development work in mind, the following figure shows the percentage of developer working schedule, spent to improving the user interface of the software/application **[A21]**.

Figure 11: Percentage of developers spending their time improving UI



Other gathered facts include:

- A majority of F/OSS developers have a high sense to respect usability bugs as real bugs (84.2%) (*asked to developers*) **[A12]**.

- Majority of the projects have completed at least one usability test (69.7%), however there are still projects which did not benefit from a usability test (*asked to both group*) [A13].
- 79.4% of the developers follow at least one guideline while designing the user interface of the application they develop (*asked to developers*) [A14].
- [A16] 43.5% of the developers first assign it to an expert in the project or get ideas from other experts. (*asked to developers*).
- [A17] 51.8% of the projects has a usability expert (*asked to developers*).
- [A18] 72.7% of the participants have read at least a chapter of book on human computer interaction (*asked to developers*). However 87.9% have read at least one article on the corresponding subject.
- [A19] 90.6% of the developers see usability as an integrated part of the development process. Only 3 developers have declared that it's an add-on property.
- [A20] Slightly more than 55.8% of the developers think that developing a user centric design requires more time, however 61.7% think that implementing a user centric design does not always require more money. On the other hand, a vast majority of the developers (87.5%) accept to work with a usability expert even if the project outcomes are delivered late just because of usability evaluation. However, roughly half (52.9%) of the developers mention that they regularly ask for inputs, suggestions and ideas from usability experts.

Another question asked to usability experts and F/OSS developers searches for methods to increase the level of communication and collaboration between usability experts and developers. Out of 78 respondents, 34 distinct answers to the question “Which methods and ways do you think will reinforce the level of communication between usability experts and developers?” are

as follows (R denotes “response”):

- **[R1]** Most effective is personal contact between developer and usability expert. Also, getting any form of user feedback, showing that the product does actually not work the way the developers thought it would.
- **[R2]** Open source software is a meritocracy. The usability experts need to be better at providing evidence to support their criticisms or suggestions, otherwise the developers have no more reason to believe them than any other person with an opinion.
- **[R3]** The project needs to try and establish that input by an usability expert is as vital to producing a new feature or interface as making it build. Usability needs to be a part of the process.
- **[R4]** Make developers watch usability testing candidates and proof that issues observed are not unique (i.e. occurred with multiple participants).
- **[R5]** All of the developers are invested in building a usable system, and respect the work of our one focused UI developer, who has a strong HCI background. This person is an actual developer, which helps immensely; she's not just a critic, but a problem-solver and builder.
- **[R6]** Have developers participate in interviewing/testing
- **[R7]** Designer need to lead projects. That's the main mechanism for propagating good design practice.
- **[R8]** Brain storming and interactive design presentation
- **[R9]** Scenarios, personas
- **[R10]** Mailing lists and bug trackers, basically.
- **[R11]** Education of UCD methods, involvement in UCD methods

- **[R12]** Pointing to various problems with existing UI/UX proposals, suggesting solutions, hinting at new UX-issues, demanding proper UX research and not just private opinions of users/mailling list members/experts.
- **[R13]** Increased respect in the validity of the field of usability by software developers and increased use of scientific usability metrics the developers can relate to.
- **[R14]** Interaction design patterns as an addition to guidelines
- **[R15]** Make it clear to developers that the usability experts help them by doing some of their UI work, thus leaving them more time for coding and implementation design.
- **[R16]** Most effective is personal contact between developer and usability expert. Also: Getting any form of user feedback, showing that the product does actually not work the way the developers thought it would.
- **[R17]** The project needs to try and establish that input by an usability expert is as vital to producing a new feature or interface as making it build. Usability needs to be a part of the process.
- **[R18]** I think the biggest problem is that "usability experts" often come on strongly as critics, rather than contributors. Part of this is perception, but part of it is the attitude of usability experts (even good ones) who don't/can't/won't develop.
- **[R19]** Have developers participate in interviewing/testing; supportive management
- **[R20]** Contextual inquiry, user-centered requirements process. Doing more to help developers understand user centered design principles, give them the immediate experience of observing product use (in the field, not the laboratory), and then having an opportunity to reflect on what they saw.

- **[R21]** mockup and prototype like crazy. Fast and out of control. Don't wait! 2. Pretty pictures mean nothing without code. Designers need to be technically savvy and engage.
- **[R22]** I like the mailing lists for this, but developers will more than likely check bug reports.
- **[R23]** Face to face meetings help in more ways than one, however that is not feasible for many projects.
- **[R24]** In developing free software for free, I never met a usability expert that was not a developer and would help for free. In developing free software for money, I work as a usability expert as well.
- **[R25]** Presence of the usability people at developer meetings; emphasis in regular communication that usability people can / should influence development too.
- **[R26]** A summit made up of project leads from developers and usability experts
- **[R27]** All of the developers have at least some training and interest in HCI/usability. We have one focused UI developer who is respected as a full member of the team, but this is at least in part because this person is a programmer, not just a "usability expert". In other words, a builder/problem-solver, not just critic.
- **[R28]** Dedicated mailing list and wiki for usability ideas.
- **[R29]** A combination of bug tracker and wiki with regular teleconferences
- **[R30]** Debates and verbal communication
- **[R31]** Better understanding of each others jobs
- **[R32]** Documentation
- **[R33]** Active listening, and testing various prototypes of multiple interfaces
- **[R34]** Strong policy in user interface change to require permission from user interface expert

4.2. Interpretation of metric data (part 1)

This section uses the outcomes from literature research and responses from survey, deriving a list of critical factors, to be used in Chapter 4 to build the table of project usability metrics. Bold letter groups defined in previous list as the quantitative survey data (**A[1] .. A[21]**) and qualitative survey data (**[R1] .. R[34]**) used where appropriate in order to strengthen the value of the inference.

4.2.1 User centered design

A user centered design process is a design philosophy in which the requirements of the user of a product are given key attention during the whole project life cycle. User-Centered Design had first mentioned with the seminal work of Norman and Draper (1986). They discuss three different types of models called 'user models' in HCI research, namely 1) the individual's own personal idiosyncratic model, 2) the generalized 'typical user' model that the designer develops to help in the formulation of the design model, or 3) the model that an intelligent program constructs of the person with which it is interacting." Since then, others (Vredenburg, 1999) have further operationalized and optimized UCD to be employed in enterprises, by deriving key components from other models (Hamel and Prahalad, 1989).

In their survey, Vredenburg et. al. (2002) find that there is a lack of measurement of UCD effectiveness and any common evaluation criteria across the industry. Over 100 respondents emphasize external objective measures but often report the use of internal and subjective measures if any measure was used at all. In another 10-question web survey conducted by Gunther, Janis and Butler (2001), the most successful activities identified by the respondents included usability testing, which was mentioned by 39% of the respondents, prototyping, and heuristic evaluation. This confirms our findings as user testing is deployed in 52.05% of the projects in our survey **[A8]**.

Gould et al (1991) claim that the user-centered design process was still not

often used in practice due to both organizational and technical reasons even though the approach had been in existence for over a decade. However, 55.5% of our usability expert respondents claim that “There's a user centered design process in my project or organization of which we follow” [A3]. This is a high percentage compared to closed source projects, however it should be noted that this question is directed only to usability experts. Therefore, this ratio is not meant to be representative for the whole set of open source projects, so is expected to lower as sample size grows.

The fact that user centered design process should be a part of a project which is carried online is also mentioned by survey respondents. In a question “Which methods and ways do you think will reinforce the level of communication between usability experts and developers?”, one of the usability experts have answered “education of UCD methods and involvement in UCD methods” [R11]. An answer from a developer mentions the usefulness of adopting user centered design process: “Doing more to help developers understand UCD principles, give them the immediate experience of observing product use (in the field, not the lab), and then having an opportunity to reflect on what they saw” [R20].

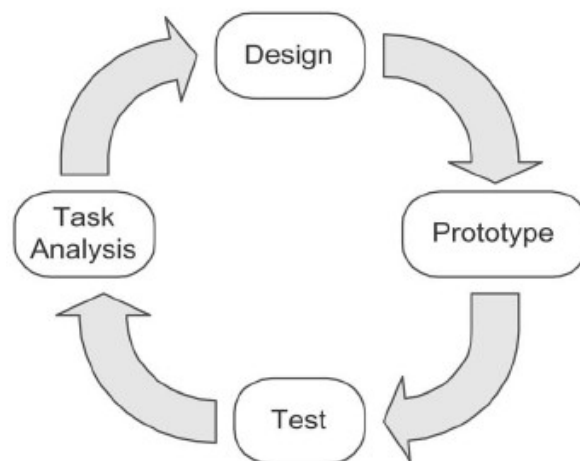
4.2.2 Developer base

There has been much successful work in developing guidelines in order to direct the design and development of user interfaces for desktop applications, which guide developers to stick with well known usability principles. According to Norman (2004), an attractive design is not necessarily the most efficient, and along with emotions, there are other points, i.e aesthetics, attractiveness and beauty. In a research, Kim & Moon (1997) claim that it is possible to manipulate the visual design factors of the customer interface in order to induce a target emotion, such as trustworthiness.

Tory and Möller (2004) stress the fact that UCD is an iterative process involving task analysis, design, prototype implementation, and testing, as

illustrated in Figure 3, as users are involved as much as possible at each design phase. Development may start at any position in the cycle, but would typically start with an analysis of the tasks the system should perform or testing of an existing system to determine its faults and limitations.

Figure 12:User centric design process



Abovementioned studies could only be done by a consistent and steady work committed on the user interface design of the application. A design itself with an expert knowledge does not validate the result, but it only shows that not only developers with little background take part in the process. In order to build a body of work, the whole usability design cycle should be investigated and deployed within the project. Product developers are often criticized for ignoring usability issues, and this is not tied to the project's development paradigm. On the other hand, while the tools that most usability experts were trained on (experimental techniques for measuring performance) were well suited for quality measurement and problem detection, they have not proved suitable for providing timely information for early design decisions, which contributed to an increased utilization of expert opinion in product development (Karat & Watson, 1993).

Participatory design is a method which encourages to change not only systems, but also the practice of collaborative system design (Schuler & Namioka, 1993; Muller & Kuhn, 1993). In a research, members of the joint

Swedish/Danish UTOPIA project worked with Nordic Graphic Workers' Union to develop an application to aid workers in page layout for newspapers, in which UTOPIA researchers used several now-iconic participatory design strategies, including low-fidelity mockups and work organization games to gain a deep, contextual understanding of users' potential interactions with new technologies while leveraging their existing skills and experiences (Ehn, 1992).

Looking at the abovementioned claims, it can be concluded that employment of usability experts with HCI background is a vital point in a F/OSS project. For example, 70.3% of the usability experts are paid by a company and/or organization [A2]. These usability experts develop interface guidelines (57.5%), test users (52.05%) and get feedback from early adopters (42.4%) [A8]. Moreover, one survey respondent have emphasized on the presence of the usability people at developer meetings; focusing on regular communication that usability people can influence development [R25]. Under the light of these findings, the following four items have been identified to act as subcriteria under the “developer base” criteria.

- Number of graphic designers
- Ratio of graphic designers to total number of developers
- Employment of usability expert(s) with HCI background
- The ratio of usability experts to total number of developers

4.2.3 Bug tracking

Many open source projects use a bug tracker software, which serve as a coordinating and collaborating tool for project management, enabling developers to track the ongoing status of bugs, wishlists and feature lists. Open source projects can be seen as a candidate for a successful online cooperation benefiting from bug tracking applications. Such applications may be simple, allowing the contributors to do a range of tasks, from simply entering a bug report to modifying the priority and categorization of individual

reports. Bug tracking is an important sign of activity for a F/OSS project, since there have been numerous researches on bug tracker data (Crowston, Howison and Annabi, 2006; Crowston, Wei, Li and Howison, 2006) to analyse the activity level of a particular project. According to Chauncey (2001), the number of usability bugs often makes up 20 to 50 percent of the overall bugs and this figure can be a powerful metric for those who say that usability is not a major issue.

For the survey, we have explicitly asked questions regarding usability bugs, and found out that more than half (66.1%) of the developers and HCI experts answering the survey benefit from Bugzilla, a well-known usability reporting tool **[A4]**. While the outcomes of deploying a bug reporting tool for the project is not examined (which is out of scope of this research), it can be concluded that F/OSS projects intensively take advantage of approaching to track bugs on the fly. However, few projects (28.3%) have a specific component dedicated to usability inside the bug tracker **[A9]**, a situation which has the potential to lead to issues like inability to track usability bugs and pitfalls during a search in bug database. On the other hand, there's a high awareness among developers regarding to consider usability bugs as real bugs (84.2%) **[A12]**. One developer also mentions that mailing lists and bug trackers are two crucial media for communication purposes **[R10]**. One reason to mention bug trackers and include in the metric table is that developers more likely check bug reports using bug trackers **[R22]**.

As Zhao states (2007), it's evident that if no formal usability testing is performed, open source projects would still lack effective mechanisms to discover usability problems (i.e error prevention) encountered by typical or in experienced users. In order to quantitatively measure the performance of the project by means of the ratio of usability bugs to total bugs, following two criteria have been defined for metric table:

1. Ratio of usability bugs to total bugs
2. Ratio of fixed usability bugs to total usability bugs

Under the light of these information, we can conclude that a successful F/OSS project needs a bug tracker with a proper configuration, allowing the users to define and prioritize usability bugs.

4.2.4 Project web page

Collaborative and cooperative software development is the core idea behind open source phenomenon. Robillard & Robillard (2000) benefit from a case study to model four types of collaborative tasks in software engineering. Deek & McHugh (2003) and DeFranco-Tommarello & Deek (2002) explain and summarize potential groupware technologies used in collaborative software development and computer supported collaboration environments. There is a lack of using advanced technology in open source projects. Currently, open source development is achieved by using lean electronic media, moderately supporting spontaneous work (Yamauchi et al., 2000). Lakhani and Hippel (2003) also explored in detail how certain open source projects benefit from collaboration tools which connect project beneficiaries

Survey respondents also point out the importance of a clean communication network among project users. For discussing usability issues, 92.5% of usability experts and developers use at least one type of media **[A5]**. Since mailing list is the most popular among them, use of this particular media will increase the level of discussion, leveraging the communication efficiency.

Many methods and collaborative environments were proposed for better communication and collaboration between developers and users (DeBellis & Haapala, 1995, Keil & Carmel, 1995). While it's evident that collaboration tools give a performance increase in almost all projects which need them, Zhao (2007) claims that there are enough reasons to believe that for open source development, particularly the development activities that users get involved, it is inappropriate to copy and paste groupware and collaborative mechanisms designed for traditional software development.

We have shown that there are many issues and solid barriers of communication between usability experts and developers. There's a clear

boundary between these entities apart from the blurry links between developer and user. It can be concluded that a web page should have a project web site, but more importantly, it needs to deploy connective links between users and developers. Providing a content management system and communication tools for project stakeholders is a helping tool to ensure a positive cooperation and information flow.

4.2.5 Tasks and guidelines

A user interface guideline is a specific set of rules in which developers adhere in order to build standardized user interface to provide familiar look and feel, comply with requirements, provide consistency and streamline development process. Ardito et al (2006) define a usability guideline using a set of suggestions from a user study. They argue that it's important “.. to provide both synchronous and asynchronous communication tools” and “.. provide communication mechanisms”. Notess (2001) highlights that specifying a user profile, a set of tasks and context of use is the requirement to measure and report usability metrics as task completion time and rate, error rate, and user satisfaction. On the other hand, user interface guidelines can also be used as heuristics by experts, in order to measure the conformity of a particular user interface against a set of well defined items. Many authors have highlighted the limitations of guidelines, standards and checklists (Reed, et al., 1999; Bevan and MacLeod, 1994). Therefore, it's vital not to depend on a user interface guideline, standard or checklist entirely and completely, but rather use them effectively where required in order to decrease the risk of limiting innovation.

According to survey output, 79.4% of the developers follow at least one guideline while designing the user interface of the application they develop **[A14]**. In order to increase the level of communication and collaboration between usability experts and developers, survey respondents suggested increased use of scientific usability metrics **[R13]**, deployment of contextual inquiry and user centered requirements processes **[R20]**, education of user centered design methods **[R11]** and developers to participate in interviewing

and testing [R6].

With the help of abovementioned deductions, we can conclude that a user profile, a proper guideline and critical user tasks list should be defined and elaborated for each and every product, and made visible to project stakeholders. Moreover, before presenting the end product to end users, a conceptual modeling is also required.

4.2.6 Accessibility, internationalization and localization

The supply of universal software has played an important part in the promotion of globalization and evidence has shown that the application of a standardized interface posed usability problems for certain ethnic groups, as the origins of its metaphors and visualization were largely foreign (Marcus, 2001). Internationalization focuses on globalizing the design process or making it accessible to cross-cultural target audiences (Aykin, 2005). The point of internationalization is to eliminate culture (cultural symbols, religious references, etc), making the product one that can be used by all or a universal design (Horton, 2005). For this reason, internationalization (i18n) allows user designers and programmers create homogenous user interfaces across product to be usable across cultures.

In a case study, Evers (2002) investigates cultural applicability of user evaluation methods, with findings suggesting some user evaluation methods are less applicable than others are for a culturally diverse user base. According to this research, cultural localization involves testing interfaces in target user cultures and conducting requirements analysis. It has been widely acknowledged that established design approaches with its standards, rules and guidelines, falls short with respect to issues relating to the cultural context, and cross-cultural user research has shown a significant difference in results by testing individuals from various socio-cultural backgrounds on accuracy and duration of task performance, user experience and user satisfaction (Shen, Wooley and Prior, 2005). Designers need to learn how different users from distinct cultures interact with interfaces, and benefit from

cultural interaction models in order to build culture-friendly interfaces. For an open source project, this requires a methodology and a list of guidelines to aid developers build cultural interfaces. If the goal of the project is to localize, then the design specifications are specialized and culture-specific (Young, 2008). It can be claimed that open source project processes do not differ from proprietary software projects. Given that F/OSS projects benefit from a wide selection of end users and developers, it should be relatively easier to receive UI feedback from a multitude of cultures.

According to the survey and literature data gathered above, it can be concluded that i) Accessibility management (i.e project web site has support to assist disabled users, and deploys a method for disabled people to reach menu items in a convenient manner) and ii) Internationalization and localization management (project web site provides a facility and a well defined method to assist translators to submit their work to project repository) subcriteria have been selected as two indicators for this metric.

4.3. Derivation of a metric model (part 2)

The following table is generated using literature review, findings from usability measurement framework, outcome of survey and heuristic deductions. Table is mainly divided into two columns, the list of criteria and source of information. Each major table row (i.e criteria or category) has its own subcategories. The final table, including a type of scale where necessary will be drawn in the next chapter.

Table 10: Recommended usability metric table for distributed and open source software projects

	Literature review	Survey findings	Heuristic deductions
1. Category: User centered design			
1.1. Mentioning “usability” in project aims	●	●	○

1.2. Heuristic evaluations defined in project tasks	○	●	○
1.3. Usability tests defined in project tasks	○	●	○
2. Category: Developer base			
2.2. Number of graphics designers	●	●	●
2.3. Ratio of graphic designers to total number of developers	●	●	●
2.4. Employment of usability expert(s) ^o with HCI background		●	●
2.5. The ratio of usability experts to total ^o number of developers		●	●
3. Category: Bug tracking			
3.1. Usability related bugs in bug tracking system	●	●	○
3.2. Usability bug prioritization	●	●	○
3.3. Usability bugs defined	●	●	○
3.4. Ratio of usability bugs to total bugs	○	○	●
3.5. Ratio of fixed usability bugs to total usability bugs	○	○	●
4. Category: Project web site			
4.1. Providing a content management system	●	●	○
4.2. Providing communication tools	●	●	○
5. Category: Tasks and guidelines			
5.1. Providing a definition of user profiles	●	●	○
5.2. Providing user interface guidelines	●	●	○
5.3. Definition of critical user tasks	○	○	●
5.4. Deployment of conceptual modeling	○	○	●
6. Category: Accessibility, i18n and l10n			
6.5. Accessibility management	●	●	○
6.6. Internationalization and localization management	○	○	●

Table 11: Symbols used to illustrate the effect on definition of the indicator

Legend icon	Description
●	Implies that source of information is used to deduce that the depicted indicator is a part of usability metric table. Parenthesis next to the filled circle shows the chapter in which this deduction first occurs.
○	Implies that information source has no effect on the definition of the indicator.
◐	Implies that information source has moderate effect on the definition of the indicator.

4.4. Final metric model table (part 3)

Table 12: Project usability metric criteria suggested by literature review

Indicator	Definition	Type of scale
Category: User centered design		
Mentioning “usability” in project aims	The aim of the project includes the importance of usability and user centric design either in project mission, vision or elsewhere on web page	<i>Nominal (yes or no)</i> May include a note to support this decision.
Heuristic evaluations defined in project tasks	Project management define details of a set of heuristic evaluations of product in a clear manner	<i>Nominal (yes or no)</i> Completeness of the evaluation documents should be defined.
Usability tests defined in project tasks	Usability tests are defined with exact timing and end user tasks together with who's in charge of tests	<i>Nominal (yes or no)</i> Completeness of the usability test guidelines and user tasks should be defined.
Category: Developer base		
Definition of user profiles	1. A user profile is defined according to each and every product developed and released at least once within the project 2. A set of user and organizational requirements is prepared	<i>Nominal (yes or no)</i> Optionally, supporting Likert scale for completeness rate. Size and order of Likert scale is defined by evaluator.

	3. Context of use is specified	
Number of graphics designers	The total number of part time and full time designers in charge of icon, banner, logo and web site design	<i>Absolute (number)</i> Number of both part time and full time designers should be mentioned.
Ratio of graphic designers to total number of developers	The ratio of graphic designers taking part in the project (both part time and full time) to total number of developers who has access to project repository	<i>Absolute (number)</i>
Employment of usability expert(s) with HCI background	The total number of part time and full time HCI experts in charge of interface usability, tests, evaluations and reporting	<i>Absolute (number)</i> The number of both part time and full time usability experts should be mentioned.
The ratio of usability experts to total number of developers	The ratio of HCI experts taking part in the project (both part time and full time) to total number of developers who has access to project repository	<i>Absolute (number)</i>
Category: Bug tracking		
Usability related bugs in bug tracking system	1. Bug database has a section defined explicitly for usability related bugs 2. This section is identified in bug tracker guidelines for users and developers issuing bugs	1. <i>Nominal (yes or no)</i> 2. <i>Nominal (yes or no)</i> May include a note to support this decision
Usability bug prioritization	1. Bug database is configured accordingly so that usability bugs can be prioritized using a severity scale 2. This severity scale is well defined	1. <i>Nominal (yes or no)</i> 2. <i>Nominal (yes or no)</i>
Usability bugs defined	Number of total usability bugs defined in a time frame	<i>Absolute (number)</i>
Ratio of usability bugs to total bugs	Ratio of total number of usability bugs to number of total bugs in a time frame	<i>Absolute (number, $R1 \in [0,1]$)</i>
Ratio of fixed	Ratio of total number of usability	<i>Absolute (number, $R2$)</i>

usability bugs to total usability bugs	bugs fixed to number of all usability bugs in a time frame	$\in [0, 1]$
Category: Project web site		
Content management system	Project web site benefits from a content management system	<i>Nominal (yes or no)</i> May include a note to support this decision.
Navigation	Project web site has a structure in which users can easily see where they are, also knowing what each page is for	<i>Ordinal (5-scale Likert: strongly agree, agree, neutral, disagree, strongly disagree)</i> May include a note to support this decision.
Category: Tasks and guidelines		
User profile	A user profile for each and every product is defined and visible within the project	<i>Nominal (yes or no)</i> May include a note to support this decision.
User interface guidelines	A guideline defining the user interface characteristics for developers is accessible from project web page	<i>Nominal (yes or no)</i> May include a note to support this decision.
Critical user tasks are defined	For each software module or application, critical user tasks are defined and documented	<i>Nominal (yes or no)</i> May include a note to support this decision.
Conceptual modeling	The product is served to end users before presenting by using a prototype	<i>Nominal (yes or no)</i> May include a note to support this decision since this may not be a strict requirement for each release.
Category: Accessibility, internationalization and localization		
Accessibility management	1. Project web site has support to assist disabled users 2. Web site uses a method for disabled people to reach menu items in a convenient manner	1. <i>Nominal (yes or no)</i> May include a note to support this decision, like lacking features). 2. <i>Nominal (yes or no)</i> May include a note to support this

		decision, like lacking features.
Internationalization and localization management	1. Project site provides a facility and a well defined method to assist translators to submit their work to project repository	1. <i>Nominal (yes or no)</i> May include a note note to support this decision.

4.5. Testing the model

Model validation is substantiating that the model, within its domain of applicability, behaves with satisfactory accuracy consistent with the model objectives (Balci, 1998). In order to provide a viable model, abovementioned metric table should be validated against well known projects which have a usability track record. For this reason, two F/OSS projects have been selected as candidates for which the metric table is filled.

1. **K Desktop Environment (KDE)**: This project started in 1998 by Matthias Ettrich in a response to build a usable and modular desktop environment for Unix and Unix-like systems (i.e Linux). KDE has grown steadily since then, and many Linux distributions ship this window manager and series of applications as either main desktop or second alternative.
2. **OpenOffice.org**: Sun Microsystems has acquired StarOffice in 1998 and decided to open its source code, providing end users a complete open source office suite. In many reviews, OpenOffice.org has been found to be equivalent to Microsoft Office in terms of usability and functionality.

Each of the following chapters refer to the two projects chosen above.

KDE project:

KDE project has 7 active usability team members as of January, 2008. There are also graphics designers and artists who prepare icons, draw UI diagrams and provide realistic mockups for the project. Moreover, artists create splash screens and themes application identity in order to increase branding. KDE

artists provide a set of guidelines on the internet, e.g artwork guidelines, splash screen guide, theme guide and icon guidelines, which help upcoming team members to understand and adopt the internals of the project. Since the project has a long-term vision, there have been times as project members join and leave. Therefore it's convenient to fix a specific month of a year for both projects, in order to be able to compare projects in terms of their usability readiness. 4 persons identified for KDE graphics artists are Everaldo Coelho (Crystal icon theme), Frank Bauer, David Vignoni (Nuvola theme) and James Richard Tyrer (hicolor icon theme). Many splash screens are already designed by the applications, and background wallpapers are usually imported from previous versions, therefore it could be concluded that main artistic activity is mainly among icon designers.

KDE has a “Usability Project”, which is an initiative to apply usability principles and practices to KDE. Goals are to help educate developers about usability in the development cycle, provide usability feedback about software through user tests and interface reviews, and promote KDE Usability throughout the Open Source Software community. Project has provided the community many research areas, i.e personas for some of the applications, usability tests of music player (Amarok), visual designer (Kivio), and download manager (Kget), heuristic evaluations (i.e image context tags, checkboxes for toggable menu items) and usability inspections of Kpdf (PDF viewer).

Another usability activity seen in KDE is bug hunting seasons. For example, in the scope of KDE 4 usability review cycle, KDE HCI working group announced the HIG hunting season, which is an experiment to include the community into the search for obvious infringements of the KDE HIG. This procedure is also backed by individual projects, and applications are analyzed by end users, by reporting user interface and interaction issues that can be stated like bugs. Users are asked to find points which disturb a seamless use experience, such as inconsistencies among applications, incomplete keyboard access, missing feedback, or overloaded configuration

dialogs, toolbars or menus. This way, applications can be transparently analyzed and tested by anyone using a given checklist early before the UI becomes mature. Usability bugs are reported directly in the bug tracking system tagged with the attribute “HIG”, so that it could later be easy for developers to search and fix a list of tagged bugs during the review cycle.

According to the data gathered from KDE bug database, KDE web page and usability project group, the following metric table including indicators and scales can be depicted.

Table 13: KDE project usability metrics

Indicator	Scale
Category: User centered design	
Mentioning “usability” in project aims	Yes. KDE has a human computer interaction user group.
Heuristic evaluations defined in project tasks	No. Heuristic evaluations are not defined in project.
Usability tests defined in project tasks	Yes. Usability tests are among tasks of KDE Usability Group.
Category: Developer base	
Number of graphics designers	4
Ratio of graphic designers to total number of developers	0.19%. Total number of developers in KDE project is 2074 (according to SVN statistics as of February 6, 2009).
Employment of usability expert(s) with HCI background	7
The ratio of usability experts to total number of developers	0.33%. Total number of developers in KDE project is 2074 (according to SVN statistics as of February 6, 2009).
Category: Bug tracking	
Usability related bugs in bug tracking system	789. Number of bugs including at least one of the phrases "usability" "human computer interface", "interface design", "design", "user centric design" as of September 2008.
Usability bug prioritization	No. Usability bugs are not prioritized.
Usability bugs defined	Yes. Users and developers are encouraged to use the word "usability" while entering a bug report.
Ratio of usability bugs to total bugs	%2.4. KDE database includes 32400 bugs as of 9

	September, 2008.
Ratio of fixed usability bugs to total usability bugs	66%. A total of 519 bugs have been fixed among 789 usability bugs.
Category: Project web page	
Content management system	Yes. Project web page benefits from a content management system.
Navigation	Likert: 5. KDE project uses an effective web navigation scheme.
Category: Tasks and guidelines	
User profile	No. Since KDE is composed of separate projects having no visible user profile, it's not possible to talk about a global and general user profile for whole project.
User interface guidelines	Yes. Developers can access to KDE Human Interface Guidelines via usability.kde.org web page.
Critical user tasks are defined	Yes. Most of the KDE applications have a corresponding user task.
Conceptual modeling	Yes. Before the introduction of the product, a prototype is developed and end users are encouraged to give feedback via blogs, mailing lists and forums.
Category: Accessibility, internationalization and localization	
Accessibility management	1. No. Project web page doesn't have a support mechanism for disabled. 2. No. Web page doesn't have a mechanism for disabled people to reach menu items effectively.
Internationalization and localization management	Yes. Project web page provides an infrastructure for translators to submit their translations.

OpenOffice.org:

Sun Microsystems has been supporting and sponsoring usability activities in OpenOffice.org, i.e usability testing and user interface design. Recently, in order to continue their effort for the entire development community, UX (user experience project) has been started with the aim “to make OpenOffice.org the best office suite in terms of usability, productivity and enjoyment”, according to project web page (User Experience Project, 2008). OpenOffice.org development also benefits from UI Design Project and Specification Project, and these 3 projects have a strict connection with each

other to provide a means of cross-sectional activity. As of September, 2008, there are 190 UX team members, however it's not known how many of them work in usability and interaction design domain.

According to Loehman (2008), "Users of OpenOffice.org complain about its cumbersome and outdated graphical user interface (GUI). A great deal of functionality is hidden in many overstuffed toolbars, poorly structured menus and complex dialogs. Functions are thus difficult to access for novice users or too inefficient to use for expert users. In addition, the GUI offers an antiquated look & feel which is hardly capable to communicate innovation".

According to the data gathered from OpenOffice.org bug database and web page, the following metric table including indicators and scales can be depicted.

Table 14: OpenOffice.org project usability metrics

Indicator	Scale
Category: User centered design	
Mentioning "usability" in project aims	Yes. One of the aims of the project is to increase the usability of the OpenOffice.org. There's also a user group dedicated to usability of the product.
Heuristic evaluations defined in project tasks	No. Heuristic evaluations do not take part in project tasks.
Usability tests defined in project tasks	No. No usability tests are defined in project tasks.
Category: Developer base	
Number of graphics designers	4. While number of part-time and full-time graphic designers change in time, usually there are 4 people working on this subject, specifically on logo design, splash screen and other visual designs.
Ratio of graphic designers to total number of developers	4/50. According to a study by Michael Meeks, number of active OpenOffice.org developers (writing 100+ lines of code every month) in 2008 is 50.
Employment of usability expert(s) with HCI background	8. Total number of developers working on interface usability, testing and evaluation is 52. 8 of them gave significant contribution to the project effectively during the last 3 months of 2008 ³ .

3 Number of experts have been taken from

http://wiki.services.openoffice.org/wiki/User_Experience_Community

developers	
Category: Bug tracking	
Usability related bugs in bug tracking system	1207. Number of bugs including at least one of the phrases "usability" "human computer interface", "interface design", "design", "user centric design" as of September 2008.
Usability bug prioritization	No. Bug database does not include a prioritization scale for usability bugs.
Usability bugs defined	Yes. Users and developers are encouraged to use the word "usability" while entering a bug report.
Ratio of usability bugs to total bugs	%1.98. OpenOffice.org database includes 61039 bugs as of 18 September, 2008.
Ratio of fixed usability bugs to total usability bugs	32.72%. A total of 395 bugs have been fixed among 1207 usability bugs.
Category: Project web site	
Content management system	Yes. Project web page benefits from Collab.Net content management system.
Navigation	Likert:5. Users can see where they are during the navigation.
Category: Tasks and guidelines	
User profile	Yes. A user profile is defined for OpenOffice.org
User interface guidelines	Yes. Developers can access a user interface guideline from OpenOffice.org UX (user experience) web page.
Critical user tasks are defined	Yes. User tasks are defined for all OpenOffice.org applications.
Conceptual modeling	Yes. Before final presentation and delivery, the product is modeled with the help of developers, using Flash or Java user interface.
Category: Accessibility, internationalization and localization	
Accessibility management	1. No. Project web page doesn't have a support mechanism for disabled. 2. No. Web page doesn't have a mechanism for disabled people to reach menu items effectively.
Internationalization and localization management	Yes. Project web page provides an infrastructure for translators to submit their translations.

5. Conclusions and implications

5.1. Introduction

This concluding chapter of the dissertation is divided in the following four sections:

1. In Chapter 5.2, a set of conclusions about research problem is identified, also main aim and findings are discussed to show the contribution to the body of knowledge.
2. In Chapter 5.3, particular focus and scope of this work are discussed to show limitations and issues of the present research.
3. In Chapter 5.4, potential research possibilities that could further yield improvements in usability, F/OSS and measurement domain is mentioned.

5.2. Conclusions about research problem

This thesis started with the observation that there's a problem with the usability of F/OSS projects with many dimensions. Since many organizations depend entirely or partially on F/OSS projects, a way to increase the usability, hence the effectiveness and efficiency of F/OSS products is a must. The aim of this thesis is to study the aspects of usability improvement and propose a metric model claiming that high usability can be ensured by first measuring and analyzing the usability readiness of a F/OSS project. Based on exploratory research, a model is derived using literature research and survey findings.

The key challenges and questions for this thesis research are summarized as follows:

1. How can we formulate the design and process activity for a collaborative project in order to benefit from usability and user centered design?
2. How can we derive a suitable measurement model for software

projects developed online in order to measure the usability readiness of the project?

3. Is it possible to use this metric to measure the degree of usability of the project?
4. What are the key technical and social challenges for sustaining the user centric design of a F/OSS project?

Individual projects may enhance the charts for their situations or have additional charts for the categories. It's also suggested that these metrics are taken in several time frames to enable inner comparison. For example, for any F/OSS project, a table can be used for delta and accumulative changes to calculate the total number of usability bugs fixed in a time frame. The major drawback of timed calculations is that, if the number of developers or usability experts are defined in the calculation as a parameter, then it's virtually hard, if not impossible, to extract this information from project web site.

The open and debatable nature of F/OSS development process shows the decentralized and low cost advancement which gives rise to a room for a chaotic management. For example, in contrast to paid developers hired to work on GNOME and KDE, volunteers do not attend community events with any regularity. Combined with paid developers' concentration on critical, infrastructural modules of the development, this leads to the conclusion that the division of labor in terms of contributors' primary areas of activity is associated with employment status and specifically with employment directly related to projects (Berdou, 2007). The thesis also shows that ensuring usability is and will be a key problem in the future releases of many projects because of this voluntary and complex nature of F/OSS projects. The key contribution to the body of knowledge of this thesis is building a metric model, which defines the criteria affecting a project's usability domain. In this model, the following items have been identified (Chapter 3.3 gives detailed explanations for each item):

1. **User centered design:** Shows how project workers defend and absorb the user centric design methodologies of the project.
2. **Developer base:** Draws a picture of how project benefits from HCI experts, by including metrics of user profile definitions.
3. **Bug tracking:** Introduces a means of how bug reporting tool in use is exploited to support developers and track design progress and support decisions.
4. **Web site:** Provides a means to assess different aspects of the project web site.
5. **Tasks and guidelines:** Gives an overview of if and how tasks and guidelines (i.e user interface guidelines, usability test templates and tutorials, guides for icon design etc) are provided for developers.
6. **Accessibility, internationalization and localization:** Informs how project benefits from a framework for disabled users, translators and localizers.

One of the main insights afforded by this study is that there could be no single and summated usability metric, as indicated in Chapter 2.7. As Zuse (2008) and Nielsen (1996) claim, assignment of simple numbers to hypothesis without knowing the empirical evidence is a major mistake, therefore it may be somewhat simplistic to assume that usability can be measured by a single number.

5.3. Limitations and issues

In this section I attempt to identify points that require further clarification. There are some issues which should be addressed before this framework can be counted as a valid method to measure the usability of software projects, given below.

1. **Project stage:** The measurement model introduced does not consider different project stages (i.e startup, mature) and considers that all

potential F/OSS projects can be measured without pointing out a requirement for maturity.

2. **Choosing a model:** In our model, a static approach is followed, in which estimated coefficients of elements are based on heuristic deductions. In contrast, there's another approach, whose parameters are specific to the product, called a dynamic model. A dynamic model allows each scale to be defined by project stakeholders, or generally saying, the person who composes the matrix table. In order to solve this issue, PUMA application can be tailored to suit the needs of different project types, by adjusting the quantitative values in template files. This way, it's possible to redefine the indicators, their descriptions and type of scale given in table 10.
3. **Weight factors:** Similar to the consideration above, the recommended weight factors could be defined in order to best guessimate for a group of F/OSS projects. How to find out which set of weight factors to use in the measurement model is still to be investigated.
4. **Survey limitations and issues:** While extreme care has been taken to take a representative respondent group (as discussed in Chapter 3.5: Survey planning on page 51), it's still open to debate whether the survey outcomes can be repeated without a major change in the results. If, for any reason, repetitive results are to be found different, then this will directly affect the data analysis chapter, rendering a different table for the metric model.

5.4. Further research

This study shares a collaborative work between different domains, i.e usability, F/OSS and measurement. While the survey purports to investigate the communication patterns between F/OSS project stakeholders, several directions for possible future research can be derived from this study.

- One possible direction of research could be to study projects with multiple maturities according to their usability readiness. This will allow different scales and factors applied to subcriteria, yielding diversified set of filled-in matrix tables for each project.
- Using a tool to analyze collected data, i.e benefiting from data mining technologies to analyze data from different perspectives and summarizing into useful information. This way, usability information gathered from different projects could be converted into knowledge and give more insight about how we could leverage the usability readiness of a given project.
- Findings from thesis suggest that the usability can be ensured by employing domain experts, i.e usability practitioners with a background in user centric design. However, the question as to when and how the projects could benefit from these experts and whether there are conflicting conditions between stakeholders of the project and the experts deserves to be investigated in detail.
- The main argument performed in this presentation is directly related to F/OSS projects, producing a software. It needs further investigation whether the idea could also be directed to other virtual groups which produce a closed source software.

We also have left the validation methodology to future work, as it requires considerable time and effort, and more importantly, inclusion of different stakeholders. In our case, a validation for this metric model should provide a quantitative measure that characterizes the agreement between predictions and observations. However there are no unified validation metric that's widely accepted as of yet.

Apart from the specific, concrete research areas depicted above, the following questions need to be better answered in the lights of a separate, exploratory research, which is out of scope of this thesis:

1. How usability in F/OSS is ensured in practice?
2. How do F/OSS developers and usability experts share their working principles?

6. APPENDIX 1: SURVEY QUESTIONS

Survey for usability experts

Number of respondents (raw results): 42

Number of respondents (net results): 3

Questions

1. Do you work as a usability expert in any F/OSS projects? (i.e are you a team member?)

- Yes: 27
- No: 9

2. Are you being paid by a company / organization because you work as a usability expert or UI designer?

- Yes: 19
- No: 17

3. Is there a user centered design process your project/organization follows?

- Yes: 20
- No: 11
- N/A: 5

4. Which bug reporting tool do you use for reporting bugs?

- Bugzilla: 19
- None: 3
- N/A: 5

- Depends: 1
- Dreamweaver: 2
- Tracback: 3
- Jira: 4
- Launchpad: 2
- Issue tracker: 2
- Trac: 1

5. In your project, which media do you use most for discussing usability issues?

- Mailing lists: 19
- IRC: 5
- Instant messenger: 4
- Bug tracker: 3
- N/A: 4
- No response: 1

6. The bug tracker we use is an effective tool in terms of usability reporting

Scale	0	1	2	3	4	5
# of responses	3	2	10	10	9	2

Where 0: strongly disagree and 4: strongly agree and 5:N/A

7. What is the focus group of your application (you may select one or more choices)

- End user: 26
- Experienced user: 25

- Casual user: 16
- Developer: 9
- Researcher: 9
- Not yet decided: 0
- N/A: 0

8. Which usability evaluation methods do you use in your project? (choose all that apply)

- Expert review: 28
- Heuristic evaluation: 18
- Interface guidelines: 22
- Getting feedback from early adopters: 22
- Paper prototyping: 16
- Focus groups: 7
- User testing: 19
- N/A (does not need a method): 0

9. There is a specific component dedicated to usability inside the bug tracker we use

- Yes: 11
- No: 22
- N/A: 3
- No response: 0

10. I know people who cannot submit bugs because of the usability problems found in the bug tracker

- Yes: 12

- No: 18
- N/A: 6

11. I've deployed surveys asking users' opinions regarding application usability (web surveys, phone surveys, face to face surveys etc).

- Yes: 22
- No: 10
- N/A: 4

12. I've experienced problems trying to describe usability issues I find on the application

- Yes: 5
- No: 22
- N/A: 2
- No response: 7

13. We have conducted at least one usability test and/or study on our application UIs.

- Yes: 26
- No: 5
- N/A: 5

14. I've experienced problems trying to convince other developers that a specific usability problem would effect users

- Yes: 26
- No: 10

15. Which methods and ways do you think will reinforce the level of communication between usability experts and developers?

- Most effective is personal contact between developer and usability

expert. Also: Getting any form of user feedback, showing that the product does actually not work the way the developers thought it would.

- Open source software is a meritocracy. The usability experts need to be better at providing evidence to support their criticisms or suggestions, otherwise the developers have no more reason to believe them than any other person with an opinion.
- personal meetings proper project management with usability integrated right from the beginning when planning a feature
- The project needs to try and establish that input by an usability expert is as vital to producing a new feature or interface as making it build. Usability needs to be a part of the process.
- Make developers watch usability testing candidates and proof that issues observed are not unique (i.e. occurred with multiple participants).
- Increased respect in the validity of the field of usability by software developers and increased use of scientific usability metrics the developers can relate to.
- interaction design patterns as an addition to guidelines
- Make it clear to developers that the usability experts help them by doing some of their UI work, thus leaving them more time for coding and implementation design.
- All of the developers are invested in building a usable system, and respect the work of our one focused UI developer, who has a strong HCI background. This person is an actual developer, which helps immensely; she's not just a critic, but a problem-solver and builder.
- - strong leadership - proper research on usability - designing early mockups/ paper prototyping - clear pathways for development process

[UX team > specification > developers > initial implementation > real life testing]

- Have developers participate in interviewing/testing
- Involving developers in contextual inquiry.
- Designer need to lead projects. That's the main mechanism for propagating good design practice.
- brain storming and interactive design presentation
- scenarios, personas
- Mailing lists and bug trackers, basically.
- Education of UCD methods, involvement in UCD methods

16. I've experienced problems trying to describe usability issues I find on the application

- Yes: 20
- No: 16

17. (If you answered yes in previous question): These problems usually resulted from the fact that developers have no background in major usability principles.

Scale	1	2	3	4	5
# of responses	2	4	2	9	3

Where 1=strongly disagree, 5=strongly agree

18. (If you answered yes in previous question) These problems usually resulted from the fact that developers lack an understanding of user requirements

Scale	1	2	3	4	5
# of responses	0	6	2	5	7

Where 1=strongly disagree, 5=strongly agree

19. I've experienced problems trying to convince other developers that a specific usability problem would effect users

- Yes: 28
- No: 8

20. If the bug reporter could attach a video file into the bug reporting tool, this would increase the handling level of bug reports

Scale	0	1	2	3	4	N/A
# of responses	2	3	8	13	7	3

Where 0: strongly disagree and 4: strongly agree

21. If the bug reporter could attach an image file into the bug reporting tool, this would increase the handling level of bug reports

Scale	0	1	2	3	4	N/A
# of responses	3	0	1	2	12	8

Where 0: strongly disagree and 4: strongly agree

22. If the bug reporter could attach an audio file into the bug reporting tool, this would increase the handling level of bug reports

Scale	0	1	2	3	4	N/A
# of responses	1	8	20	6	1	0

Where 0: strongly disagree and 4: strongly agree

23. What (if any) are the negative effects of your presence in mailing list as a

usability expert?

- Asking for usability-driven changes at the wrong time (e.g. no time capacity from the developers) can cause some irritations :)
- Less experienced members of the usability community expect the usability pro's to lead the project and to take ownership for decisions, etc., instead of being welcome to actively participate. Developers focus too much on specific people (i.e. the pro's).
- You get to fight about issues pretty often.
- Too many e-mails
- It is hard for some to realize they are wrong and doing stupid. Then they go all defense mode and spikes up.
- Regardless of role, one needs to build trust in the OS project before throwing one's weight around... build trust over time, then pounce.
- none, if anything it is because i am a usability expert for fun and do not have the time focus full effort on helping
- none, if anything it is because i am a usability expert for fun and do not have the time focus full effort on helping
- I don't know, I might get off topic sometimes
- None (3)

24. What (if any) are the positive effects of your presence in mailing list as a usability expert?

- Increase of quality in use of the application.
- I suppose my usability opinions do carry a little more weight than other people's.
- Give reasoning why a design idea by developers is good or bad
- Clear role within the project - people know whom to contact in case of

questions.

- Encouraging discussion of issues.
- If reports about usability bugs appear on the ML, the developers oftentimes make me aware of that and trust in my assessments.
- We can make a better product. :)
- Pointing to various problems with existing UI/UX proposals - suggesting solutions - hinting at new UX-issues - demanding proper UX research and not just private opinions of users/ mailing list members/experts
- Not sure
- Good visibility for usability and helps foster better relationships with developers.
- Perhaps at least some do listen and understand?
- Duh... better design!
- Once the team figured out that I was 'trustworthy' they always asked my opinion
- I'm not sure - you would have to ask them.
- Sometimes people will call me out specifically to address or answer a question
- We can modify some of the not so good implementations by the developers.

25. Which methods and ways do you think will reinforce the level of communication between usability experts and developers?

- Most effective is personal contact btw developer and usability expert. Also: Getting any form of user feedback, showing that the product does actually not work the way the developers thought it would. (Same as no. 11)

- Real-time discussion integrated with an online-whiteboard and vnc to any of the participants' desktop personal meetings
- The project needs to try and establish that input by an usability expert is as vital to producing a new feature or interface as making it build. Usability needs to be a part of the process.
- I think the biggest problem is that "usability experts" often come on strongly as critics, rather than contributors. Part of this is perception, but part of it is the attitude of usability experts (even good ones) who don't/can't/won't develop.
- Proper UX-research: real life experiments - strong leadership - clear development pathways
- Have developers participate in interviewing/testing; supportive management
- More shared design meetings.
- Contextual inquiry, user-centered requirements process. Doing more to help developers understand UCD principles, give them the immediate experience of observing product use (in the field, not the lab), and then having an opportunity to reflect on what they saw.
- I'd recommend applying a cattle prod into anus. Seriously, in my experience anyone who does programming just plain deserves that.
- 1. mockup and prototype like crazy. Fast and out of control. Don't wait!
2. Pretty pictures mean nothing without code. Designers need to be technically savvy and engage.
- I like the mailing lists for this, but developers will more than likely check bug reports.
- Face to Face meetings help in more ways than one, however that is not feasible for many projects

26. Do you have any plans for gathering usability requirements, building UI

guidelines or refactoring and/or amending product UI in your project?

- Yes, and we published it in our web page: 14
- Yes, but not published: 18
- No: 4

Survey for developers

Number of respondents (raw results): 43

Number of respondents (net results): 40

Questions

1. The project I work for is funded by a company

- Yes: 16
- No: 24

2. The project I work for has a mailing list dedicated to usability discussions

- Yes: 22
- No: 18

3. Which bug reporting tool do you use for reporting bugs?

- Bugzilla: 24
- Gnats: 0
- Mantis: 4
- Jira: 2
- None: 5
- Launchpad: 3
- Trac: 4
- Berlios: 1
- Gforge: 1
- Scarab: 1
- Issuezilla: 1
- N/A (does not require): 1

4. The bug tracker we use is an effective tool in terms of usability reporting

Scale	0	1	2	3	4	5
# of responses	2	3	6	9	14	6

Where 0: strongly disagree and 4: strongly agree, 5:N/A

5. I've received at least one complaint from a user who cannot submit bugs because of the usability problems found in the bug tracker in my project

- Yes: 14
- No: 25
- N/A (i.e a bug tracker is not used): 1

6. In your project, which media do you use most for discussing usability issues?

- Mailing lists: 24
- IRC: 7
- Instant messenger: 1
- Bug tracker: 4
- N/A: 4

7. There is a specific component dedicated to usability inside the bug tracker we use

- Yes: 10
- No: 28
- N/A: 2

8. I respond to usability bugs as real bugs in the bug tracking tool we use (out of 39 responses)

- Yes: 32
- No: 4

- N/A: 2
- No answer: 1

9. When I receive a usability bug, I first assign it to an expert in my team or get her ideas

- Yes: 17
- No: 14
- N/A: 8
- No answer: 1

10. We have conducted at least one usability test and/or study on our application UIs.

- Yes: 20
- No: 15
- N/A: 2
- No answer: 3

11. We have deployed surveys asking users' opinions regarding application usability (web surveys, phone surveys, face to face surveys etc).

- Yes: 13
- No: 22
- N/A: 3
- No answer: 2

12. What is the focus group of your application? (you may select one or more choices)

- End user: 30
- Experienced user: 22
- Casual user: 12

- Developer: 14
- Researcher: 12
- System administrator: 2
- Not yet decided: 0
- N/A: 0

13. We follow a/any guideline while developing the user interface of our application

- Yes: 27
- No: 7
- N/A: 6

14: Which methods and ways do you think will reinforce the level of communication between usability experts and developers?

- I am a developer and a usability expert. In developing free software for free, I never met a usability expert that was not a developer and would help for free. In developing free software for money, I work as a usability expert as well.
- More meetings!
- Don't know
- Presence of the usability people at developer meetings; emphasis in regular communication that usability people can / should influence development too.
- Face to face meetings are essential.
- A summit made up of project leads from developers and usability experts
- All of the developers have at least some training and interest in HCI/usability. We have one focused UI developer who is respected as

a full member of the team, but this is at least in part because this person is a programmer, not just a "usability expert". In other words, a builder/problem-solver, not just critic.

- Dedicated mailing list and wiki for usability ideas.
- Testing, reporting, correcting, testing... mailing list
- We've read about usability but there is no usability experts on the team.
- A combination of bug tracker and wiki with regular teleconferences
- Debates and verbal communication
- Better understanding of each others jobs
- Documentation
- Waterfall model
- Active listening, and testing varies prototypes of multiple interfaces
- Strong policy in UI change to require permission from UI expert

15. Is there a usability expert in your project?

- Yes: 14
- No: 13
- N/A: 13

16. Which usability evaluation methods do you use in your project? (choose all that apply)

- Expert review: 23
- Heuristic evaluation: 11
- Interface guidelines: 20
- Getting feedback from early adopters: 19
- Paper prototyping: 8

- Focus groups: 9
- User testing: 21
- N/A (i.e our project does not need an evaluation method): 1
- No answer: 2

17. I've advocated my project's usability vision at least once in the past to project developers and/or users

- Yes: 14
- No: 7
- No answer: 19

18. In my opinion, implementing a user centric design always requires more time

- Yes: 19
- No: 15
- No answer: 6

19. In my opinion, implementing a user centric design always requires more money

- Yes: 13
- No: 21
- No answer: 6

20. I've experienced problems trying to describe usability issues I find on the application at least one in the past

- Yes: 21
- No: 13
- No answer: 6

21. I've experienced problems trying to convince other developers that a

specific usability problem would effect users

- Yes: 22
- No: 12
- No answer: 6

22. If answered “yes” in previous question,

These problems usually resulted from the fact that developers have no background in major usability principles:

Scale	0	1	2	3	4	5
# of responses	0	1	4	9	7	1

Where 0: Strongly disagree, 4: Strongly agree , 5: N/A

23. These problems usually resulted from the fact that developers lack an understanding of user requirements

Scale	0	1	2	3	4	5
# of responses	0	1	3	4	12	2

Where 0: Strongly disagree, 4: Strongly agree , 5: N/A

24. Do you regularly ask for inputs, suggestions and ideas from usability experts?

- Yes: 18
- No: 16
- No answer: 6

25. Would you accept to work with a usability expert even your project outcomes are delivered late just because of usability evaluation?

- Yes: 35

- No: 5

26. I personally see usability as

- Add-on property: 3
- Integrated part of the development process: 29
- Other: 0
- No answer: 8

27. I've read at least one book or a chapter of a book on human computer interaction

- Yes: 24
- No: 9
- No answer: 7

28. I've read at least one article (printed or online) on human computer interaction

- Yes: 29
- No: 4
- No answer: 7

29. Keeping your weekly project development work in mind, I spend percent of my working schedule improving the user interface of the software/application I maintain

- Less than 10%: 2
- 10% to 30%: 9
- 30% to 50%: 14
- 50% to 70%: 6
- More than 70%: 1
- N/A: 2

- No answer: 6

30. If the bug reporter could attach a video file into the bug reporting tool, this would increase the handling level of bug reports

Scale	0	1	2	3	4	5
# of responses	1	3	4	4	15	7

Where 0: strongly disagree, 4: strongly agree, 5: N/A

No answer: 6

31. If the bug reporter could attach an image file into the bug reporting tool, this would increase the handling level of bug reports

Scale	0	1	2	3	4	5
# of responses	2	1	1	5	13	12

Where 0: strongly disagree, 4: strongly agree, 5: N/A

No answer: 6

32. If the bug reporter could attach an audio file into the bug reporting tool, this would increase the handling level of bug reports

Scale	0	1	2	3	4	5
# of responses	2	9	10	11	2	0

Where 0: strongly disagree, 4: strongly agree, 5: N/A

No answer: 6

REFERENCES

1. OpenSUSE (2007). *Better desktop project*. Retrieved July 12, 2007 from betterdesktop.org.
2. Relevantive (2007). *Linux usability report*. Retrieved July 5, 2007 from www.relevantive.de.
3. What is the balanced scorecard? (2007). Retrieved July 8, 2007 from <http://www.balancedscorecard.org/basics/bsc1.html>.
4. Abran, A., Surya, W., Khelifi, A., Rilling, J., Seffah, A., Robert, F. (2003). *Consolidating the Usability Models*. Paper presented at 11th annual International Software Quality Management Conference.
5. Andreasen, M. S, Nielsen, H. V., Schrøder, S. O., Stage, J. (2006). *Usability in Open Source Software Development: Opinions and Practice*. Information Technology and Control, ISSN 1392 – 124X. Vol. 35, No. 3 A
6. B. W. Boehm , J. R. Brown , M. Lipow, (1976). *Quantitative evaluation of software quality*. Proceedings of the 2nd international conference on Software engineering, p.592-605, October 13-15, 1976, San Francisco, California, United States
7. Behlendorf, B. (1999). *Open source as a business strategy*. In: M. Stone, S. Ockman, and C. DiBona (editors). *Open Sources: Voices from the Open Source Revolution*. Sebastopol, Calif.: O'Reilly & Associates, pp. 149-170
8. Benson, C., Adam, E., Nickell, S., Robertson, C. Z. (2002). *GNOME Human Interface Guidelines 1.0*. Retrieved July 2, 2007 from <http://developer.gnome.org/projects/gup/hig/1.0>.
9. Bicarregui, J. (2008). *Software Engineering Methods – Measurement and testing*. Lecture notes, Imperial College and Rutherford Appleton Laboratory. Retrieved January 10, 2008 from <http://theory.doc.ic.ac.uk/~jcb1/SEIII/2000/allslides2000.ppt>

10. Bloomer, S., Croft, R. (1997). Pitching usability to your organization. ACM Interactions Magazine November-December 1997 pp. 18-26
11. Boehm, B. W., J. R. Brown, H. Kaspar, M. Lipow, G. J. MacLeod, M. J. Merritt (1973). Characteristics of Software Quality, TRW Software Series TRW-SS-R 73-09.
12. Brooke, J. (1996). SUS: a quick and dirty usability scale. In P. W. Jordan, B. Thomas, B. A. WeerdMeester & I. L. McClelland, Eds. Usability Evaluation in Industry, pp. 189 - 194. London: Taylor & Francis.
13. Chin, J. P., Diehl, V., Norman, K. L. (1988). Development of an instrument measuring user satisfaction of the human-computer interface. Human Factors in Computing Systems CHI188, pp. 213-218. New York: ACM & Addison-Wesley.
14. Crowston, K., Wei, K., Li, Q., Eseryel, U. and Howison, J. (2005). Coordination Of Free/Libre Open Source Software Development, 26th International Conference On Information Systems, Las Vegas, NV, pp. 181-193.
15. Çetin, G., Verzulli, D., Frings, S. (2006). *F/OSS usability test results report*. EU-funded tOSSad project contract no: 015981.
16. Çetin, G., Verzulli, D., Frings, S. (2007). *An Analysis of Involvement of HCI Experts in Distributed Software Development: Practical Issues*. Online Communities and Social Comput., HCII 2007, LNCS 4564, pp. 32–40. Springer-Verlag Berlin Heidelberg
17. Dillman, D. A. (2000). *Mail and Internet surveys: The tailored design method*. 2nd ed. New York: John Wiley.
18. Dumas, J., Redish, J. C. (1999). *A Practical Guide to Usability Testing*. Intellect Books, Exeter, UK, 1999
19. Eason, K. (1988). *Information technology and organisational change*. Taylor and Francis.

20. Raymond, E. (1999). *The revenge of the hackers*. In: M. Stone, S. Ockman, and C. DiBona (editors). *Open Sources: Voices from the Open Source Revolution*. Sebastopol, Calif.: O'Reilly & Associates, pp. 207-219
21. Fenton N.E, Pfleeger, S. L. (1997). *Software Metrics: A rigorous and practical approach, 2nd ed*. Boston, M.A PWS Publishing
22. Fenton, N. (1994). *Software Measurement: A necessary scientific basic*. IEEE Transactions on Software Engineering. vol. 20, no. 3, March 1994.
23. Fricker, D. R., Schonlau, M. (2002). *Advantages and Disadvantages of Internet Research Surveys: Evidence From the Literature*. *Field Methods*; 14; 347
24. Frishberg, N., Dirks, A. M., Benson, C., Nickell, S., Smith, S. (2002). *Getting to Know You: Open Source Development Meets Usability*. Extended Abstracts of the Conference on Human Factors in Computer Systems (CHI 2002). New York, NY: ACM Press, pp. 932-933.
25. Godfrey, M. W., Tu, Q. (2000). *Evolution in Open Source Software: A Case Study*. p. 131, 16th IEEE International Conference on Software Maintenance (ICSM'00)
26. Göktürk, M., Çetin, G. (2007). *Out of Box Experience Issues of Free and Open Source Software*. HCI International Proceedings Volume 1, LNCS_4550, ISBN: 978-3-540-73104-7
27. Interaction (HCI), Sun Microsystems, Inc. Retrieved 28 November, 2002 at http://developer.gnome.org/projects/gup/ut1_report
28. ISO 9241-11:1998 (1998). *Ergonomic requirements for office work with visual display terminals (VDTs) Part 11: Guidance on usability*.
29. Jeffries, R. Miller, J.R., Wharton, C., Uyeda, K. (1991). *User interface evaluation in the real world: a comparison of four techniques*. Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology, p.119-124, 1991, New Orleans, Louisiana,

United States

30. Kirakowski, J. (1994). *The use of questionnaire methods for usability assessment*. Chapter from Measures & Methods for Quality of Use, Edited by Prof. T Bösner. Unpublished book.
31. Kirakowski, J., and Corbett, M. (1993). *SUMI: The Software Usability Measurement Inventory*. British Journal of Educational Technology, 24, 210--212.
32. Lerner, J. and Tirole, J. (2002). *Some Simple Economics of Open Source*. Journal of Industrial Economics, 46 (2), 125-156.
33. Lucey, N. M. (1991). *More than Meets the I: User-Satisfaction of Computer Systems*. Unpublished thesis for Dipl. Applied Psychol, University College Cork, Ireland.
34. Lund, M. A. (1997). *Another approach to justifying the cost of usability*. ACM Interactions Magazine pp. 49-51
35. Miles, M. B., Huberman, A.M. (1985). *Qualitative Data Analysis*. Sage, New York, p.31.
36. Mockus, A., Fielding, R. T., Herbsleb, J. D. (2002). *Two Case Studies Of Open Source Software Development: Apache And Mozilla*. ACM Transactions on Software Engineering Methodology, pp. 309-346.
37. Muehling, J., Paul, C. L. (2004). *Usability and open source software*. Interfaces by British HCI Group, ISSN 1351-119X, pp.18-21
38. Neely, A. (1998). *Measuring Business Performance*. Profile Books, Ltd., London, U.K, 1998
39. Nichols, D., Twidale, M. (2002). *The usability of open source software*. Retrieved 21 April, 2008 from http://www.firstmonday.dk/issues/issue8_1/nichols
40. Nichols, D. M., Thomson, A., Kirsten, T., Yeates, S. (2001). *Usability and open-source software development*. In Elizabeth Kemp, Chris Phillips,

- Kinshuk, and John Haynes, editors, Symposium on Computer Human Interaction, pages 49–54, Palmerston North, New Zealand. ACM SIGCHI NZ.
41. Nielsen, J. (1993a). *Iterative User Interface Design*. IEEE Computer Vol. 26, No. 11. November 1993, pp. 32-41.
 42. Nielsen, J. (1993b). *Usability engineering*. Boston: Academic
 43. Nielsen, J. (1994). *Heuristic evaluation*. In J. Nielsen & R. L. Mack (Eds.), *Usability inspection methods* (pp 24-62). New York: Wiley
 44. Nielsen, J. (2008). *Usability metrics*. Reached 10 January, 2008 from <http://www.useit.com/alertbox/20010121.html>
 45. Raymond, E. S. (1999). *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly & Associates, Sebastopol, CA
 46. Reitmayr, E., “*File browser behaviour*”, retrieved July 2, 2007 from http://usability.kde.org/activity/reports/2006/report_filebrowser-screening-test_08.05.pdf
 47. Sandusky, R. J., L. Gasser, & G. Ripoché (2004). *Bug report networks: Varieties, strategies, and impacts in a F/OSS development community*. MSR 2004: International Workshop on Mining Software Repositories, ICSE 2004, IEEE International Conference on Software Engineering, Edinburgh, Scotland, UK
 48. Sauro, J., Kindlund, E. (2005). *A method to standardize usability metrics into a single score*. CHI 2005, Portland, Oregon, USA.
 49. Scacchi, W. (2002). *Understanding the requirements for developing open source software*. IEEE proceedings on software, 149(1), pp24-39.
 50. Shackel, B. (1991). *Human Factors for Informatics Usability*. Cambridge University Press, p. 24.
 51. Shneiderman, B. (1987). *Designing the User Interface: Strategies for*

Effective Human-Computer Interaction. Addison-Wesley Publishing Co., Reading, MA

52. Smith, S., Engen, D., Mankoski, A., Frishberg, N., Pedersen, N., Benson, C. (2001). *GNOME Usability Study Report*. Sun GNOME Human Computer Interaction Laboratory
53. Svetinovic, D. and Godfrey, M.M. (2005). *Software and Biological Evolution: Some Common Principles, Mechanisms, and a Definition*. In Proc. of the 8th International Workshop on Principles of Software Evolution (IWPSE05), Lisbon, Portugal.
54. Winship, D. (2007). The opinions of Dan Winship. Retrieved June 20, 2008 from <http://mail.gnome.org/archives/desktop-devel-list/2006-February/msg00115.html>
55. Van Welie, M., van der Veer, G. C., Elins, A. (1999). *Breaking Down Usability*. Proceedings of INTERACT 99 (Edinburgh) IOS Press, 613-620.
56. von Hippel, E., von Krogh, G. (2003). Open source software and the private-collective innovation model: Issues for organizational science. *Organization science* vol. 14, No: 2. March-April 2003
57. Wikipedia, Software Metric topic. Retrieved February 12, 2008 from http://en.wikipedia.org/wiki/Software_metric
58. Apache, How the Apache Software Foundation work. Retrieved June 8, 2008 from <http://www.apache.org/foundation/how-it-works.html>
59. Çetin, G., Göktürk, M. (2007). *Usability in open source community*. *ACM Interactions* volume 14, issue 6. pp. 38-40
60. Basili, V., Caldiera, G., Rombach, H. D. (1994). Goal Question Metric Approach. *Encyclopedia of Software Engineering*, pp. 528-532, John Wiley & Sons, Inc.
61. Basili, V. (2002). "Measurement Frameworks", Briefing slides from Course CMSC 735, University of Maryland, Fall 2002

62. Kirakowski, J. SUMI. The de facto industry standard evaluation questionnaire for assessing quality of use of software by end users. <http://www.ucc.ie/hfrg/questionnaires/sumi/index.html> (accessed: January 2007).
63. Seffah, A., Donyaee, M., Kline, R.B. and Padda, H.K. (2006). "Usability measurement and metrics: a consolidated model", *Software Quality Journal*, Vol. 14, pp. 159-78.
64. Rubey, R. J., Hartwick, R. D. (1968). "Quantitative measurement of program quality", *Proceedings of the 23rd National Conference*, pp. 671–677. ACM Publications
65. Murine, G. E. (1980). "Applying software quality metrics in the requirements analysis phase of a distributive system". Technical report, Metriqs Incorporated, California
66. Pfleeger, S., Rombach, D. (1994). *Measurement based process improvement*, IEEE Software.
67. Wangenheim, C. G., Punter, T., Anacleto, A. (2003). "Software Measurement for Small and Medium Enterprises", 7th International Conference on Empirical Assessment in Software Engineering (EASE), Keele, UK.
68. Rombach, H. D. (1991). "Practical benefits of Goal-Oriented Measurement", *Software Reliability and Metrics*, Elsevier Applied Science, pp. 217-235.
69. Basili V. R., Weiss, D. M. (1991). "Evaluation of a software requirements document by analysis of change data". In *Proceedings of the Fifth International Conference on Software Engineering*, pages 314–323. IEEE Computer Society Press, 1981.
70. Heberlein, T. A., Baumgartner, R. B. (1978). "Factors affecting response rates to mailed questionnaires: A quantitative analysis of the published literature." *American Sociological Review*: 43:447-462

71. Dillman, A., Sinclair, M. and Clark, J. (1993). "Effects of questionnaire length, respondent-friendly design and a difficult question on response rates for occupant-addressed census mail surveys". *Public Opinion Quarterly*: 57: 289-304.
72. Cantone, G., Donzelli P. (1998). "Production and Maintenance of Software Measurement Models", European Software Control and Metrics Conference (ESCOM-ENCRESS)
73. Briand, L. C., Morasca, S., Basili, V. R. (1994). "Goal-Driven Definition of Product Metrics Based on Properties", University of Maryland - Technical Report CS-TR-3346, 1994.
74. Zuse, H. (1991). "Software Complexity", Walter de Gruyter.
75. Briand, L. C., Differding C. M., and Rombach, H. D. (1997). "Practical Guidelines for Measurement-Based Process Improvement", *Software Process Improvement and Practice Journal*, Vol. 2.
76. Emam, K. E. (2000). "A Methodology for Validating Software Product Metrics," National Research Council of Canada, Ottawa, Ontario, Canada NCR/ERC-1076.
77. Gould, J.D., Boies, S.J., Lewis, C. (1991). "Making usable, useful, productivity-enhancing computer applications", *Communications of the ACM*, 34, 75-85.
78. Vredenburg, K., Mao, J., Smith, P. W., Carey, T. (2002). "A survey of user-centered design practice", *Proceedings of the SIGCHI conference on Human factors in computing systems: Changing our world, changing ourselves*, Minneapolis, Minnesota, USA
79. Gunther, R., Janis, J., & Butler, S. (2001). "The UCD Decision Matrix: How, When, and Where to Sell User-Centered Design into the Development Cycle", <http://www.ovostudios.com/upa2001/>, accessed 18 February, 2008.

80. Norman, D., Draper, S.W. (1986). "User-Centered System Design: New Perspectives on Human-Computer Interaction". Erlbaum, Hillsdale, NJ.
81. Vredenburg, K., (1999). "Increasing Ease of Use: Emphasizing Organizational Transformation, Process Integration, and Method Optimization". *Communications of the ACM*, 42, 67-71.
82. Hamel, G. and Prahalad, C.K., (1989). "Strategic intent". *Harvard Business Review* May-June 1989.
83. Crowston, K., Howison, J., and Annabi, H. (in press). Information systems success in free and open source software development: Theory and measures. *Software Process: Improvement and Practice* (Special Issue on Free/Open Source Software Processes)
84. Crowston, K., Wei, K., Li, Q., Howison, J. (2006). "Core and periphery in Free/Libre and Open Source software team communications", *Proceedings of the 39th Hawaii International Conference on System Sciences*.
85. Wilson, C., Coyne, K.P. (2001). "The Whiteboard: Tracking Usability Issues: To Bug or not to Bug?" *Interactions*, 8(3), 15-19.
86. Notess, M. (2001). "Usability, User Experience, and Learner Experience". Retrieved from <http://www.elearnmag.org>
87. Ardito, C., Costabile, M.F., De Marsico, M., Lanzilotti, R., Levialdi, S., Roselli, T., and Rossano, V. (2006). "An Approach to Usability Evaluation of e-Learning Applications, Universal Access in the Information Society", *International Journal*, 4 (3), 270 – 283.
88. Reed, P., Holdaway, K., Isensee, S., Buie, E., Fox, J., Williams, J. and Lund, A. (1999). "User interface guidelines and standards: progress, issues, and prospects". *Interact Comput* 12:110–142
89. Bevan, N., Macleod, M. (1994). "Usability measurement in context". *Behaviour & Information Technology* 13 1&2, 132-145.

90. Nielsen, J. (1994). "The Usability Engineering Life Cycle," *Computer*, Vol. 25, No. 3, Mar. 1994, pp. 12-22.
91. Nielsen, J., Molich, R. (1990). "Heuristic Evaluation of User Interfaces". *Proceedings of CHI Conference on Human Factors in Computing Systems*. New York: ACM, 249-256.
92. Lewis, C., Polson, P., Wharton, C., Rieman, J. (1990). "Testing a Walkthrough Methodology for Theory-Based Design of Walk-up-and-Use Interfaces". *Proceedings of CHI Conference on Human Factors in Computing Systems*. New York: ACM, 235-242.
93. Wharton, C., Bradford, J., Jeffries, R., Franzke, M. (1992). "Applying Cognitive Walkthroughs to More Complex User Interfaces: Experiences, Issues, and Recommendations". *Proceedings of CHI Conference on Human Factors in Computing Systems*. New York: ACM, 381-388.
94. Kjeldskov, J., Stage, J. (2004). "New Techniques for Usability Evaluation of Mobile Systems". *International Journal of Human-Computer Studies* (60), 599-620.
95. Pemberton, L., Griffiths, R. (2003). "Usability evaluation techniques for interactive television". *Proc. HCI International*.
96. Gauntlett, D., Hill, A. (1999). "TV Living: TV, Culture and Everyday Life". London: Routledge.
97. Daly-Jones, O. (2002). *Navigating your TV: the Usability of Electronic Programme Guides*. In *Usable iTV 1/3*, pp. 2-6.
98. Ericsson, A., & Simon, H. (1984). *Protocol analysis: verbal reports as data*. Boston: The MIT Press.
99. Del Galdo, E. M., Williges, R. C., Williges, B. H., & Wixon, D. R. (1987). *A critical incident evaluation tool for software documentation*. In L. S. Mark, J. S. Warm, & R. L. Huston (Eds.), *Ergonomics and Human Factors*. New York: Springer-Verlag, 253-258

100. Chin, J. P., Diehl, V. A., Norman, K. L. (1988). *Development of an instrument measuring user satisfaction of the human-computer interface*. Proceedings of CHI Conference on Human Factors in Computing Systems. New York: ACM, 213-218.
101. Karat, C. M. (1993). Usability Engineering in Dollars and Cents. IEEE Software, v.10 n.3 p. 88-89.
102. O'Reilly, T. (1999). Lessons from Open Source Software Development. Comm. ACM, vol. 42, no. 4, Apr. 1999, pp. 33-37.
103. Nakakoji, K., Yamamoto, Y., Kishida, K., Ye, Y. (2002). Evolution patterns of open-source software systems and communities. In The International Workshop on Principles of Software Evolution, Orlando Florida, 2002.
104. Healy, K., Schussman, A. (2003). The Ecology of Open-Source Software Development, University of Arizona . Working Paper (www.kieranhealy.org)
105. Sullivan, L.P., (1986). Quality function deployment. Quality Progress 19 (6), 39–50.
106. Sarkis, J., Liles, D.H. (1995). Using IDEF and QFD to develop an organizational decision support methodology for the strategic justification of computer-integrated technologies. International Journal of Project Management 13 (3), 177–185.
107. Nibbelke, R., Ferro, D., Hoogeboom, P. (2001). Design and evaluation with the human in mind. Air and Space Europe 3(3–4), 218–220.
108. Chang, C.H., Lin, J.T. (1991). Data flow model of a total service quality management system. Computers and Industrial Engineering 21 (1–4), 117–121.
109. Tan, K.C., Xie, M., Chia, E., 1998. Quality function deployment and its use in designing information technology systems. International Journal of

Quality and Reliability Management 15 (6), 634–645.

110. ISO (1998). Ergonomic requirements for office work with visual display terminals (VDTs) -- Part 13: User guidance. Retrieved May 31, 2008 from <http://www.iso.org>
111. Godfrey, M., Tu, M. (2001). Growth, Evolution, and Structural Change in Open Source Software. In Proceedings of the 4th International Workshop on Principles of Software Evolution. ACM Press, 2001: 103-106.
112. Robles, G., Amor, J. J., Gonzalez-Barahona, J. M., and Herraiz, I. (2005). Evolution and Growth in Large Libre Software Projects. In Proceedings of the Eighth International Workshop on Principles of Software Evolution (IWPSE 2005). IEEE Computer Society, 2005: 165-174.
113. Scacchi, W. (2006). Understanding Open Source Software Evolution. In Software Evolution and Feedback. John Wiley & Sons
114. Koch, S. (2007). Software Evolution in Open Source Projects—A Large-Scale Investigation. In Journal of Software Maintenance and Evolution: Research and Practice 2007; 19: 361-382.
115. Deshpande, A., Riehle, D. (2008). “The Total Growth of Open Source.” In Proceedings of the Fourth Conference on Open Source Systems (OSS 2008). Springer Verlag, 2008, to appear.
116. Card, S. K., Moran, T. P., Newell, A. (1993). The Psychology of Human-Computer Interaction, Hillsdale, New Jersey.
117. Byrne, M. D., Wood, S. D., Foley, J. D., Kieras, D. E., Sukaviriya, P. N. (1994). Automating interface evaluation, CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems (New York, NY, USA), ACM, pp. 232–237.
118. Butler, K. (1996). Usability engineering, Interactions 1, 58–75.
119. Nielsen, J. (1990). Big playbacks from discount usability, IEEE Software

- 3, 107–108.
120. Nielsen, J. (1996). Usability metrics: Tracking interface improvements, *IEEE Software* 6 (1996), 12–13.
 121. Zuse, H. (2008). Retrieved from <http://irb.cs.tu-berlin.de/~zuse/pub/sl-leuven.html> on June 10, 2008.
 122. Cook TD, Campbell DT. (1979). *Quasi-Experimentation: Design & Analysis Issues for Field Settings*. Boston, Mass: Houghton Mifflin Co.
 123. Ye, Y., Kishida, K. (2003). "Toward an Understanding of the Motivation of Open Source Software Developers," *icse*, p. 419, 25th International Conference on Software Engineering (ICSE'03), 2003
 124. Lakhani, K. R., Wolf, R. G. (2003) "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects" (September 2003). MIT Sloan Working Paper No. 4425-03. Available at SSRN: <http://ssrn.com/abstract=443040> or DOI: 10.2139/ssrn.443040
 125. Hertel, G., Niedner, S., Herrmann, S. (2003). "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel", *Research Policy* Volume 32, Issue 7, , Open Source Software Development, July 2003, Pages 1159-1177.
 126. Berdou, E. (2007). *Managing the Bazaar: Commercialization and peripheral participation in mature, community-led Free/Open source software projects* . PhD Thesis, London School of Economics and Political Science .
 127. Kim, J. & Moon, J.Y. (1997). Designing Towards Emotional Usability in Customer Interfaces: Trustworthiness of cyber-banking system interfaces. *Interacting with Computers*, 10, 1-29.
 128. Tory, M., Möller, T. (2004). Human Factors in Visualization Research, *IEEE Transactions on Visualization and Computer Graphics*, 10(1):72-

84.

129. Karat, J., Watson, T. J. (1993). Usability measurement – its practical value to the computer industry. Panel during Interchi'93.
130. Schuler, D., Namioka, A. (1993). Participatory Design: Principles & Practices. Lawrence Erlbaum Associates.
131. Muller, M., Kuhn, S. (1993). Special Issue on Participatory Design, CACM 36:4, June.
132. Ehn, P. (1992). Scandinavian design: On participation and skill. In PS Adler & TA Winograd (eds.) Usability: Turning technologies into tools, 96-132. Oxford University Press NY.
133. Deek, F. P., McHugh, J. A. M. (2003). Computer-Supported Collaboration with Applications to Software Development. Springer.
134. Robillard, P. N., Robillard, M. P. (2000). Types of collaborative work in software engineering. Journal of Systems and Software, Volume 53, Issue 3, Pages 219-224
135. Yamauchi, Y., Yokozawa, M., Shinohara, T., Ishida, T. (2000). Collaboration with Lean Media: How Open-Source Software Succeeds. Proc. Computer Supported Cooperative Work Conf. (CSCW 00), ACM Press, pp. 329-338.
136. Lakhani, K. R., von Hippel, E. (2003). How Open Source Software Works: 'Free' User-to-User Assistance. Research Policy 32, no. 6: 923–943.
137. DeBellis, M., Haapala, C. (1995). User-Centric Software Engineering, IEEE Expert: Intelligent Systems and Their Applications, v.10 n.1, p.34-41.
138. Keil, M., Carmel, E. Customer-developer links in software development, Communications of the ACM, v.38 n.5, p.33-44, May 1995
139. Norman, D. (2004). Emotional design: Why we love (or hate) everyday things (New York: Basic Books)

140. Marcus, A., (2001). International and intercultural user interfaces. In: Stephanidis, C. (Ed.), *User Interfaces for All: Concepts, Methods, and Tools*. Lawrence Erlbaum, Mahwah, NJ, pp. 47–63.
141. Aykin, N. (2005). Overview: where to start and what to consider. In N. Aykin (Ed.), *Usability and internationalization of information technology* (pp. 3–20). Mahwah, NJ: Lawrence Erlbaum Associates Publishers.
142. Horton, W. (2005). Graphics: the not quite universal language. In N. Aykin (Ed.), *Usability and internationalization of information technology* (pp. 157–188). Mahwah, NJ: Lawrence Erlbaum & Associates Publishers.
143. Young, P. A. (2008). Integrating culture in the design of ICTs. *British Journal of Educational Technology*, 39(1), 6–17.
144. Evers, V. (2002). Cross-cultural applicability of user evaluation methods: a case study amongst Japanese, north American, English and Dutch users. In: *Proceedings CHI 2002*, Minneapolis.
145. Zhao, L., Deek, F. P. (2004). User collaboration in open source software development. *Electronic Markets* 14 (2):89-103.
146. Shen, S.-T., Woolley, M., and Prior, S. (2005). "Towards culture-centred design," *Interacting with Computers* (In Press, Corrected Proof).
147. Carpenter, C. L., Murine, G. E. Applying software quality metrics. In *ASQC Quality Congress Transactions*, Milwaukee, Wisconsin, 1983. American Society for Quality Control.
148. User experience project (2008). User experience project on OpenOffice.org. Retrieved December 17, 2008 from <http://ux.openoffice.org/>
149. Loehman, F. (2008). State of the UX project, presented in OpenOffice.org conference, Beijing, 2008.

ÖZGEÇMİŞ

Görkem Çetin, 1974 yılında Ankara’da doğdu. ODTÜ Matematik Öğretmenliği Bölümü’nden lisans ve Bahçeşehir Üniversitesi Bilgisayar Mühendisliği Bölümü’nden yüksek lisans derecelerini aldı. TÜBİTAK, Vestel ve Turkcell firmalarında çalıştı. Yazılmış 4 Türkçe kitabı vardır.