

**towards Open Source Software adoption and dissemination  
tOSSad**

**Contract No 015981**

# **Usability Involvement in F/OSS Projects**

**A usability guideline**

**D19**

**Version 1.0**

**08.12.2006**



## Copyright

This document is Copyright © 2006 by its contributors as listed in the section titled “version control”. You can distribute it and/or modify it under the terms of either the GNU General Public License, version 2 or later (<http://www.gnu.org/licenses/gpl.html>), or the Creative Commons Attribution License, version 2.5 or later (<http://creativecommons.org>).

All trademarks within this guide belong to their legitimate owners.

## Version control

Version	Date	Author	Organisation
0.1	1 August 2006	Görkem Çetin	TUBITAK/UEKAE
0.2	2 August 2006	Leonid Ozirkovskyy	ULIBIN
0.3	1 September 2006	Damiano Verzulli	PDA
0.4	14 September 2006	Damiano Verzulli	PDA
0.5	19 September 2006	Görkem Çetin	TUBITAK/UEKAE
0.6	3 October 2006	Görkem Çetin	TUBITAK/UEKAE
0.61	6 October 2006	Sandra Frings	IAT
0.7	12 October 2006	Görkem Çetin	TUBITAK/UEKAE
0.8	20 October 2006	Al Harris	KnowNet
1.0	08 December 2006	Kaan Erkan	TUBITAK/UEKAE

## Change history

Version 0.1: Initial document

Version 0.2: Addition of chapter 3

Version 0.3: Fixed typos and other grammar issues. Restructured the whole Chapter 3.

Version 0.4: Added introduction

Version 0.5: Modified chapter 4

Version 0.6: Modified chapter 4, added abbreviations

Version 0.61: Proof reading

Version 0.7: Minor changes, addition of chapter 4.2.2

Version 0.8: Proofreading

Version 1.0: Quality check

## Release approval

Name	Role	Date
Kaan Erkan	Project Coordinator	08/12/2006

## Abbreviations

BIND:	Berkeley Internet Name Daemon
CRM:	Customer Relationships Management
CVS:	Concurrent Versions System
DNS:	Domain Name Service
F/OSS:	Free/Open Source Software
GNOME:	GNU Network Object Modeling Environment
GNU:	GNU is Not Unix
HCI:	Human Computer Interaction
IEEE:	Institute of Electrical and Electronics Engineers
KDE:	K Desktop Environment
RFC:	Request for Comment
SO-HO:	Small Office Home Office
SUS:	Single UNIX Specification
tOSSad:	Towards Open Source Software Adoption and Dissemination

## Contents

1. Executive Summary.....	5
2. Introduction.....	6
2.1. A new, unknown problem.....	7
2.2. A new, unknown target.....	8
2.3. Lack of skills.....	8
2.4. Relationship.....	9
2.5. Adoption of standard.....	9
2.5.1. Compliance of the application behavior.....	9
2.5.2. Compliance of the Development Process.....	11
2.6. Towards a better F/OSS Usability.....	12
3. Current status.....	15
3.1. Distributed nature of F/OSS development process.....	15
3.2. Concentration of F/OSS on experts and power users.....	16
3.3. Standards for software development process.....	16
3.4. Other well-known HCI and usability standards.....	18
3.5. Collaboration between usability experts and F/OSS developers.....	19
3.6. Insufficient usage of HCI guidelines and usability experts' recommendations by F/OSS developers.....	19
4. Towards a better interaction with usability experts.....	21
4.1. Collaborating with usability experts.....	22
4.2. Communication of developers and usability experts.....	23
4.2.1. Bugzilla: friend or foe?.....	24
4.2.2. Collaboration on mailing lists.....	25
4.2.3. Communication via project web page.....	25
4.2.4. Some common usability methods.....	26
5. Summary.....	28

## 1. Executive Summary

---

The EU funded project tOSSad - Towards Open Source Software Adoption and Dissemination - is a Coordination Action (CA) and its aims are to improve the outcomes of the Free / Open Source Software (F/OSS) communities throughout Europe by supporting the coordination and networking of these communities by means of state-of-the-art studies, national program initiations, usability cases, curriculum development and implementation of collaborative information portals and web based groupware.

The project workpackage “F/OSS Usability Study” deals with sharing usability aspects and requirements among usability experts in order to help developers incorporate more user-centered design in the development process of F/OSS applications.

Traditionally, F/OSS developers have focused more on the features of a specific application, most of the time ignoring the necessity of user-centric design. This has mainly stemmed from the fact that developers have little interaction with HCI (human computer interaction) studies, knowledge bases and reports. Moreover, the lack of user interface designers has resulted in a lack of awareness of this area. As a consequence, the user centric design phenomenon within F/OSS applications has been neglected.

In this report, tOSSad partners try to answer the question “How can we achieve more usability within geographically distributed projects?”. We'll first define the basic characteristics of a F/OSS project and investigate the problems encountered that slow down the usability process and F/OSS development. In the following chapter, we analyse the F/OSS developers, i.e. why their interest in usability is not inline with their interest in feature adding for a software project. Then, we give advice and recommendations on how to attract and convince usability experts to participate in F/OSS projects.

The summary part gives a brief overview of the basic findings regarding the deep involvement of usability experts in F/OSS projects.

## 2. Introduction

---

Since its origin, in 1983, the main focus of the F/OSS community has been the development of very technical and specific applications targeting engineers or, more specifically, computer scientists. This choice was mostly a forced one, as at that time, due to various reasons<sup>1</sup>, there were no F/OSS at all and, as such, the first step to take was necessarily the development of the building-blocks (compiler, linker, debugger) of the F/OSS community. Once production-ready, those building-blocks could have acted as enablers for a much wider community of programmers that, with such tools, would have been able to develop many other kinds of applications.

Throughout the 20<sup>th</sup> century the F/OSS community has been mostly involved in such development activities and as a result a very long list of excellent applications have been released. As stated above, most of those applications suffered from the same problem: they were oriented towards people with a strong technical background; they were developed by computer scientist for computer scientist.

Although there have been some attempts to simplify the usage of several F/OSS applications, nothing serious happened until 1996. It was only at that time that someone started changing the target or, better, tried to include the needs of common end users within the F/OSS community.

With the wide adoption of personal computers in the SO-HO market (Small-Office, Home-Office), the requirements for applications and related operating systems were optimally supplied by proprietary software vendors. Due to very aggressive marketing campaigns, in 1996 a huge number of end-user computers were running a proprietary operating system (Microsoft DOS or some version of Windows) together with a certain number of proprietary applications. Those applications could rely on more than 10 years of Microsoft experience regarding the “usability”: “...*the effectiveness, efficiency and satisfaction with which users can achieve tasks...*” of such operating system and applications (ISO 9241-11).

As given above, the formal definition of usability is focused on generic tasks and does not limit the target to end-users. Hence it is possible to speak about the usability level of a compiler (typically used by a computer scientist to create an executable application) as well as it is possible to speak about the usability level of a text-processor or a file-manager (mostly used by end-users to write simple letters and to store and retrieve documents within a computer).

Furthermore, the definition above does not restrict the usability concept to specific kinds of applications: usability aspects need to be applied to F/OSS applications as well as to proprietary applications.

Although the tOSSad project is focused on F/OSS in general, we consider a 360° analysis of the usability level of all the F/OSS applications is mostly unfeasible: given the development rate of current F/OSS projects, the time needed to perform such a study may well be longer than the time required by the F/OSS community to release completely new applications.

Even if a strict separation between server-side software and client-side software cannot be made, we decided to focus our study on the latter, trying to understand the dynamics of F/OSS usability when applied to client-side, desktop applications or, in other words, to applications developed and focusing on the previously mentioned end-users.

Despite the long history of the F/OSS community, with the advent of the 21<sup>st</sup> century, something new has happened within the F/OSS community. Following the announcement of

---

<sup>1</sup> Insert some reference to “Open Sources: Voices from the Open Source Revolution” and “Free Code”

the KDE project<sup>2</sup> (1996) and the GNOME project<sup>3</sup> (1997), an initial group of F/OSS developers started taking care of end-users needs and, as such, building two completely new desktop environments (KDE and GNOME) that would both have a critical role in bringing the F/OSS world to increasing numbers of office desktops.

Both KDE and GNOME had the end-users needs within their mission remit, recognising usability as a critical factor from the beginning of both projects.

It can be argued that such a consideration (usability as a key factor for a F/OSS application) was not shared by many other F/OSS projects, including both well-known projects (various compilers, for example) and/or other minor projects (the ones launched and maintained by many single developers). We really think that if such considerations apply to many F/OSS projects, they apply to many proprietary projects as well. Usability is an issue to be taken into account for all software projects, regardless of its nature (F/OSS or not). Furthermore, the more the application is going to be used by end-users, the more usability issues act as a key-factor for the applications success.

As of today, we are aware of desktop F/OSS applications developed with usability in mind as well as desktop F/OSS applications built with no care to usability. At the same time, we are aware of very usable desktop proprietary applications as well as unusable desktop proprietary applications.

Perhaps the most important difference between the F/OSS world and the proprietary one is best expressed in the history of computer desktops, with Microsoft Windows accruing an additional 13 years of experience (from 1985 to 1998) in desktop software and still, at this time, maintaining a competitive advantage in this area in respect to F/OSS alternatives (mainly KDE and GNOME). Furthermore, it is worth nothing that, due to the commercial nature of proprietary software vendors, usability was (and still is) seen as a revenues generator: the more usable the software, the more licenses sold, the more revenues generated. This issue can heavily impact the development process as companies, being driven by the business needs, will force the adoption of usability requirements even if they are often underestimated by many programmers with a more technical mindset.

The longer history of desktop proprietary applications together with several other factors, led to a current situation where the number of unusable desktop F/OSS applications is much higher than the number of unusable desktop proprietary applications.

In the following paragraphs we present the reasons that led to such a lack of usability by many desktop F/OSS applications.

### 2.1. A new, unknown problem

Trying to identify the starting date of the usability-problem-solving within the F/OSS community, we can consider October 14<sup>th</sup> 1996 as such date. It was exactly at that time that Matthias Ettrich, a young Dutch university student, sent a sort of “call for programmers” in order to start the KDE project. As clearly reported in his message<sup>4</sup>, end-users were going to be the main focus of the KDE project:

---

2 [www.kde.org](http://www.kde.org)

3 [www.gnome.org](http://www.gnome.org)

4 <http://www.kde.org/documentation/posting.txt>

*“...The idea is NOT to create a GUI for the complete UNIX-system or the System-Administrator. [...]. The idea is to create a GUI for an ENDUSER. Somebody who wants to browse the web with Linux, write some letters and play some nice games....”*

*Matthias Ettrich – KDE project founder – October, 14<sup>th</sup> 1996*

Before KDE the F/OSS community mostly dealt with technical software whose requirements and specifications were totally defined by relying on the various skills available within the community. With the launch of the KDE project, for the first time, the F/OSS community focused on end-users. Communications with such users, the understanding of their needs, their approach to computer usage and lots of other issues were completely new problems that the F/OSS community needed to face.

After 21 months from its launch, on July 12<sup>th</sup> 1998, the KDE project released the first version: KDE 1.0.

## 2.2. A new, unknown target

Desktop F/OSS applications are, by definition, to be used on a desktop computer and generally this means that most users dealing with such software present some common set of ICT skills, neither expert nor novice but somewhere between. Nevertheless, since its beginning, the F/OSS community had always and explicitly focused on computer scientists. This assumption is clearly shown in the original “call for programmer” for the GNU project that Richard Stallman, founder of the Free Software Foundation, posted on the Internet on September 27<sup>th</sup> 1983. Quoting such post<sup>5</sup>:

*“...To begin with, GNU will be a kernel plus all the utilities needed to write and run C programs: editor, shell, C compiler, linker, assembler, and a few other things. After this we will add a text formatter, a YACC, an Empire game, a spreadsheet, and hundreds of other things. We hope to supply, eventually, everything useful that normally comes with a Unix system, and anything else useful, including on-line and hardcopy documentation....”*

*Richard Stallman – Free Software Foundation – September, 27<sup>th</sup> 1983*

Even nowadays, more than 20 years later, this paragraph sounds very tricky even to the technical reader.

In the 13 years between the launch of the GNU (1983) and KDE (1996) projects, F/OSS software has exclusively been developed by computer scientist for computer scientist.

As such, KDE represented a milestone, forcing the F/OSS community to focus on end-users, a completely new target needing appropriate rules and to be correctly approached and managed.

## 2.3. Lack of skills

Since the early days, the F/OSS community has been able to develop a huge amount of software relying exclusively on the skills available within the community itself. Mostly, every member had a strong computer background and, as such, it was easy to scratch down the internal details of compilers, editors, linkers, kernels and all the other software

<sup>5</sup> <http://www.gnu.org/gnu/initial-announcement.html>

typically used and needed by them. In other words, the F/OSS community was able not only to define the requirements but also to develop their preferred applications.

With the advent of F/OSS desktop computing the situation quickly changed and a completely new set of skills was needed. Interface design, visual communication and information architecture became mandatory concepts for a winning desktop environment, concepts mostly unknown by the F/OSS community member.

## 2.4. Relationship

It is common for computer programmers to prefer a single-developer scenario where he can make all the decisions alone even in respect to work in a big project where external coordination is needed. Many programmers quickly recognised that the group approach was the only approach offering an optimal way to succeed in the developing complex F/OSS applications..

It was relatively easy to put F/OSS programmer around the same table as they were motivated by the very same principle: freedom. Freedom to access and change the code, freedom to redistribute the code and freedom to use the code in their preferred way.

Several other communities, dealing with information architecture and system design concepts existed and, as in the F/OSS community, they were motivated by similar principles.

Raising desktop F/OSS needs required the establishment of new links between such communities but unfortunately this was not an easy task to achieve: F/OSS community was not used to involving external members within its own projects and, also, external communities did not recognise and/or share the fundamental reasons at the base of the F/OSS movement.

Nowadays, eight years after the launch of the KDE project, we are still faced with this problem even if the interaction between communities has increased substantially.

## 2.5. Adoption of standard

When regarding the compliance to standards by the F/OSS community, such compliance can be seen from two different perceptions:

- compliance of the application behavior;
- compliance of the development process adopted by the F/OSS development community.

### 2.5.1. Compliance of the application behavior

In terms of compliance to existing standards the exponential growth of the Internet has been a key-enabler for all the F/OSS community by both increasing the interactions within the community and, also, by enabling the community access to the technical details of the Internet itself.

With all the technical specifications available on the Internet, everyone is able to start developing F/OSS applications fully compliant with such specifications (also called RFC - Request for Comments<sup>6</sup>). Without the requirement to pay any kind of fees, even the smallest and/or poorest programmer is absolutely able to write his own compliant F/OSS program. Furthermore, the compliance to RFCs acted as a warranty in terms of interoperability between the developed program and the rest of services available on the

---

6 [http://en.wikipedia.org/wiki/Request\\_for\\_Comments](http://en.wikipedia.org/wiki/Request_for_Comments)

Internet.

The growth of the Internet has also hugely increased the number of developers, allowing the creation of smaller subsets of developer communities where each community shares exactly the same needs and desires. As a consequence, the number of F/OSS applications rapidly increased and the applications became increasingly mature and powerful.

As a proof of the above assertions, we see that the Internet is heavily based on F/OSS applications. Critical systems like DNS and other very important services like the web and e-mail are running on top of F/OSS operating systems (GNU/Linux, xBSD systems and others) and are serviced by F/OSS applications (BIND for DNS; Apache for web; Sendmail and Postfix for e-mail). Those and many other applications have clearly demonstrated that, given an open availability of the standards, compliance to those standards and given the freedom of the Internet community to act without external interferences, F/OSS applications can be far superior to their proprietary counterparts.

We do think that the openness of standards is a key point. Speaking about application compliance, it is useful to briefly discuss POSIX<sup>7</sup>, a standard dealing with the characteristics that every Unix systems should have and, as such, whose compliance is a key point also for the Linux Kernel, the very core of each GNU/Linux system, that Linus Torvalds<sup>8</sup> started to develop as a FLOSS project in 1991. POSIX was defined by the Institute of Electrical and Electronics Engineers (IEEE)...

*"...during late 1980s in an attempt to create a common API – Application Program Interface – for all Unix systems. At that time no one trusted any of the Unix vendors, the IEEE shepherded the standards process and created the 1003 series of standards, known as POSIX. [...] Few people have actually seen an official POSIX standard document, as the IEEE charges money for copies. Back before the Web became really popular, I bought one just to take a look at the real thing. It wasn't cheap (a few hundred dollars, as I recall). Amusingly enough, I don't think Linus Torvalds ever read or referred to it when he was creating Linux; he used other vendors' references to it and manpage descriptions of what POSIX calls were supposed to do."*

*Jeremy Allison<sup>9</sup> – Open Source 2.0 – A Tale of Two Standard*

The different approach to standard definition followed by RFCs and POSIX is evident: from one side the open nature and free-availability of RFCs has been fundamental to create applications that nowadays manage a huge number of critical servers and services all around the Internet.

From the other side (POSIX); the closed nature of the standard has built a still active barrier that seriously impacts the development of any F/OSS application, even some of the most important ones, including the Linux Kernel.

Fortunately enough in 1998, the POSIX standard became superseded by SUS - Single Unix Specification -, and quoting again Jeremy Allison<sup>10</sup>:

<sup>7</sup> <http://en.wikipedia.org/wiki/POSIX>

<sup>8</sup> [http://en.wikipedia.org/wiki/Linus\\_Torvalds](http://en.wikipedia.org/wiki/Linus_Torvalds)

<sup>9</sup> Jeremy Allison is one of the lead developers of the Samba Team. He handles the coordination of Samba development world wide. He works for Novell, which funds him to work full time on improving Samba and solving the problems of Windows and Linux interoperability.

<sup>10</sup> [http://samba.org/samba/news/articles/low\\_point/tale\\_two\\_stds\\_os2.html](http://samba.org/samba/news/articles/low_point/tale_two_stds_os2.html)

*“SUS is...adopted by all the major Unix vendors and shepherded by the Unix standards body, 'the Open Group'. It was a great leap forward when this specification was finally made available for free on the Web from the Open Group web site at <http://www.unix.org>.”*

*Jeremy Allison – Open Source 2.0 – A Tale of Two Standard*

Furthermore, it is worth noting that despite the, not full compliance to POSIX, the overall quality of the Linux Kernel and its wide adoption has let the Linux interfaces take over as the more important standard for Unix-like systems to follow, in some ways supplanting POSIX and SUS.

### **2.5.2. Compliance of the Development Process**

The process of proprietary software development is regulated by a wide set<sup>11</sup> of standards. These standards take into account usability requirements in an early stage of the development process, during both the software requirement specification and the design stage, and enforce the development process to go through a testing stage during which all of the usability requirements need to be verified and validated.

Testing the usability level of proprietary software is frequently carried out by independent companies. Regarding the development process of F/OSS applications, the situation is opposite. Although it could be possible to apply the standards above also to the F/OSS development process, it has to be said that such standards are proprietary and for this reason not available on a free model. Hence, F/OSS developers need to pay to access such documents and while this could be easy for developers and projects financially supported by some third-parties, it is mostly unfeasible for the bigger mass of developers that act alone and/or are involved in projects that cannot rely on any kind of funding.

The development of any large application has always been a very complex activity requiring some level of coordination (the larger the application, the higher the coordination level) and technical background. Although in the early days of computing the art of programming was performed in the so-called “hackerdom”<sup>12</sup>, out of any business logic, the rise of proprietary software vendors quickly changed the overall scenario.

If, in the early days, the F/OSS community was organised only as a “Bazaar style” model with no strict rules and/or formal hierarchy<sup>13</sup>, when commercial companies entered the software development market, they quickly felt the needs to formally organise the whole cycle of development activities. It was based on such needs that various specifications began to appear. In the late '90s, IEEE released several official standards whose content are common practice in several software houses still nowadays. Unfortunately, those standards have been, and still are, generally unavailable to many developers as they cannot be accessed via the web on a free basis (they are charged at a set amount, depending on the standard).

It is easy to recognise the difference between the F/OSS community approach and the proprietary vendors approach: where the F/OSS community prefers to deal with open standards like RFCs and with no involvement of any external non-F/OSS-oriented organisation like IEEE or ISO, proprietary vendors follow a very different route that involves such official standard organisation.

11 [http://standards.ieee.org/catalog/olis/arch\\_se.html](http://standards.ieee.org/catalog/olis/arch_se.html)

12 <http://www.oreilly.com/catalog/opensources/book/raymond.html>  
<http://www.oreilly.com/openbook/freedom/>

13 <http://www.catb.org/~esr/writings/cathedral-bazaar/>

To the F/OSS community, the result of such an approach is that, with the exception of a few bigger F/OSS projects, all the development activities carried out by the F/OSS community do not follow IEEE or ISO rules at all but, instead, are centered around self-defined rules and organisations.

## 2.6. Towards a better F/OSS Usability

How can the usability level of desktop F/OSS applications be increased? This is the question that we have to answer and, ultimately, the goal of the tOSSad workpackage “F/OSS Usability Study”. Unfortunately, finding an answer is not an easy task.

We think that the usability problem can be approached by combining several different factors:

- **creating a distributed development, involvement, evaluation and reporting methodology for F/OSS usability:** While the F/OSS community is more comfortable acting on a very distributed base and with a very low number of physical interactions between developers, this is not the case for other communities, including the usability communities. Even if web, email, forums and several other Internet technologies are properly used by not-so-skilled people, we think that a lot can still be improved in the area of on-line collaboration. F/OSS community could support the usability communities in such an improvement approach providing them with platforms and training that can make the on-line collaboration easier. Improving the efficiency of on-line activities is a key factor in lowering the ratio between the number of desktop F/OSS projects and the total amount of available usability skills, by increasing the number and the utility of the professionals involved whilst minimising the overhead of internal communication management. During the last 20 years, F/OSS developers have deeply analysed and optimised exactly these last two points.

Nevertheless, a critical factor for a proper involvement of usability experts in F/OSS development, is sharing and following the base principles of F/OSS community. After all, both F/OSS developers and usability experts feels satisfied not only by their monetary remuneration but, mainly, by the self-satisfaction they feel once their creativity has been fully expressed through their respective projects. A F/OSS project grants a usability professional licence to express his creativity as much as he wants.

- **taking care of usability requirements since the very beginning of F/OSS projects:** We already mentioned that developers typically prefer to work alone. Even in cases where little groups are formed around a certain F/OSS project, it is common that such groups include only F/OSS developers and, as such, usability issues are not properly considered. Sometimes, it is only during the development activities that usability issues arise and, again, in some cases, attempts to search for usability expertise are made. The result is the already mentioned poor usability level for most of the desktop F/OSS applications.

In order to increase the overall usability level of desktop F/OSS applications, this approach needs to be changed: all programmers know that application requirements need to be formally defined before the start of the developing activity, in the same way usability requirements need to be specified and defined well before the development activity. Developers need to understand that usability has to be one of their first priorities, from the very beginnings of the project, and that usability is not “just another feature” that can simply be added during the project.

- **maintaining, or increasing, the current trend in desktop F/OSS development:** We have already mentioned the history of desktop computing and emphasised the longer tradition of proprietary operating systems (Microsoft Windows) in respect to F/OSS desktop environments (KDE and GNOME).

## Introduction

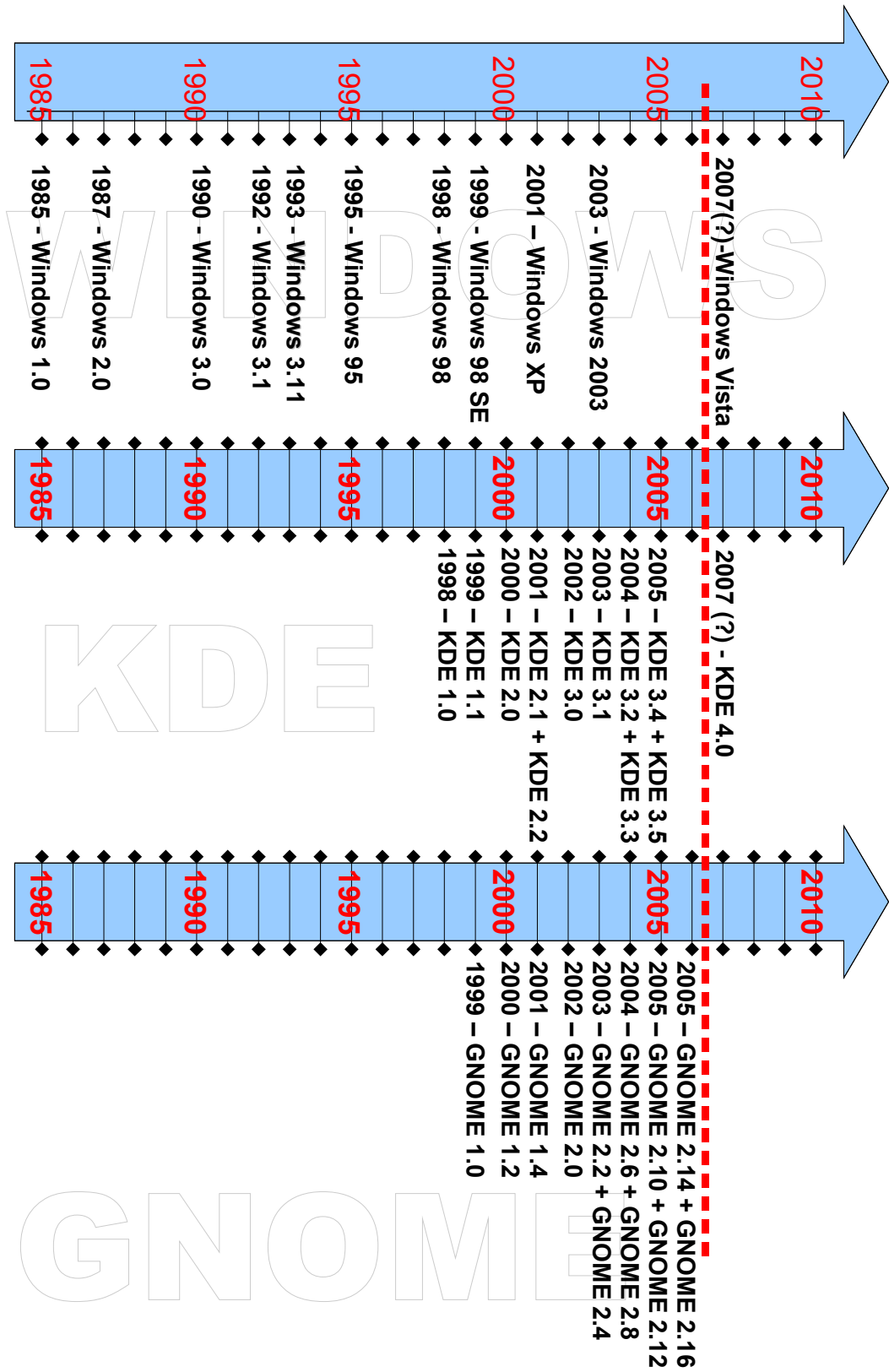
A brief analysis of the development trends, quickly reported in Figure 1, clearly shows some interesting items: while the development of new Microsoft Windows<sup>14</sup> release seems to follow a regular path and, furthermore, seems to have slowed during the last three years, the development of both KDE and GNOME releases have been scheduled along a much tighter time frame.

Moreover, comparing each release of both KDE and GNOME with the previous one, it turns out that the progress of both projects is clearly shown in the level of usability. Should such progress continue for the future releases, it is easy to expect a very comfortable and usable KDE 5 and GNOME 4.

---

<sup>14</sup> Here we consider only the version of Windows oriented to desktop" usage.

Figure 2: History timeline of KDE, GNOME and MS Windows



### 3. Current status

---

According to some experts<sup>15</sup>, the majority of end-users rely on proprietary software for their daily activities, not looking at alternatives provided by F/OSS software, and without knowing that F/OSS software has a reputation of reliability, efficiency and functionality.

Most end-users base such an approach on the fact that F/OSS applications are relatively technically sophisticated.

When deeply investigated, it comes out that the majority of F/OSS applications present an unsatisfactory usability level.

The reasons for such a poor usability level can be investigated in terms of:

- Differences in the organisation of the software development process between the F/OSS community and proprietary companies;
- Attitude toward the compliance to standards during the software development activities;
- Degree of involvement of external experts in the development process;
- Features and nature of the F/OSS community.

Some of the factors above result from the intrinsic structure of the software development process adopted by the F/OSS community. Other factors result from the failure, by F/OSS software developers, to take into account the qualifications, the psychological features and the requirements of end-users within the SOHO (Small Office/Home Office) segment.

#### 3.1. Distributed nature of F/OSS development process

One of the essential factors which causes an unsatisfactory usability of several F/OSS applications is the distributed nature of F/OSS projects<sup>16</sup>. It is common, for F/OSS developers, to work in the same project yet be from different geographical regions, different countries and even different continents.

For such an heterogeneous base, it is difficult to adhere to uniform requirements and/or set of rules: when a F/OSS developer feels the need of a new feature he can add it to the software, and if it gets checked-in by the project owners, it will be there for all to use. Even in the case where project owners refuse to include the proposed improvement within the main project code, chances are that the F/OSS developer will be free to start a brand new project, managed directly by himself and with the new feature included (the term "fork" is used to describe such an action).

The obvious problem with this model is the now well known scourge of "creeping features": too many features and options begin to overwhelm and overshadow the core functionality of the software<sup>17</sup> and will worsen its usability.

A good design of the application user interface could greatly improve the usability level, especially if such a design is combined with the definition of a set of default application options that a user will not be forced to learn before using the application.

This approach is very different from the one that focuses only on the power of the application to deliver a huge number of complex features, a huge set of options and a minimal interface.

<sup>15</sup> [http://www.firstmonday.org/issues/issue8\\_1/nichols/](http://www.firstmonday.org/issues/issue8_1/nichols/)

<sup>16</sup> [http://www.rashmisinha.com/archives/05\\_04/open-source.html](http://www.rashmisinha.com/archives/05_04/open-source.html)

<sup>17</sup> <http://actsofvolition.com/archives/2004/april/theriseof>

## 3.2. Concentration of F/OSS on experts and power users

Traditionally the F/OSS software development has been conducted by the F/OSS community itself, relying completely on the skills available within the community or, in only a few cases, involving external experts. As such the user interfaces found in most F/OSS applications have been developed by engineers for engineers.

Only recently, mainly thanks to the F/OSS projects like OpenOffice.org, Firefox, KDE, GNOME and other major projects focused on the end-user of the SO-HO market, has the feedback cycle with conventional users started to be taken into account.

Considering the resources provided by such major projects, the F/OSS community still lacks usability skills and, furthermore, does not register enough connection with the usability expert community.

Actually, some authors<sup>18</sup> consider that it is in the F/OSS movement's fundamental interest to search, find and urge contributions from the usability professionals, as this is essential to further increase the usability level of the F/OSS application and, as such, to gain market share.

## 3.3. Standards for software development process

If analysed in terms of compliance to software development standards, the current development process of F/OSS applications is expected to be not so different to current proprietary applications.

If major proprietary software houses like Microsoft, Oracle or Symantec are expected to strictly follow the various ISO and/or IEEE standard throughout all the development cycle of their products, it is also expected that the development process of current major F/OSS applications like OpenOffice.org, Firefox, KDE, GNOME is also led by a little team of developers that enforce the strict compliance with at least the major standards.

Things were much different in the early days of the F/OSS movement. The development of the Linux kernel, for example, started in the early '90s as a kind of game from its creator, Linus Torvalds, and as such it was built with no particular attention to any standard at all.

Also within the Free Software Foundation, since the early '80s, Richard Stallman put huge efforts into assuring the absolutely strict compliance of the developed software to standards like POSIX, but little is known about the development process itself. Such processes were surely conducted without significant reference to any of the standards available at that time.

As of today, the compliance to standards are key issues in software certification, an environment where such compliances need to be formally proved by a software house in order to have its software certified as compatible with another software or fit for purpose within certain specified environments. This is, for example, typical in the development process of hardware drivers to be used within an operating system.

Nevertheless it has to be said that the certification issue is typical of very big applications; at the other end, if we focus on little projects, we can surely say that each developer follows his own rules regardless of the type of software (proprietary vs. F/OSS).

As cited above, the proprietary nature of the main standards (e.g. IEEE, ISO, etc.) and the need to support direct costs to access them, results in developers that not only do not comply with these standards, but also do not completely know their requirements. This, again, is regardless of the type of the software developed.

With all above paragraphs in mind, it could be useful to get a quick overview of the current

<sup>18</sup>Professional Usability in Open Source <http://www.mprove.de/script/04/chi/>

standards dealing with software development. Due to the proprietary nature of such standards, we deliberately avoid showing any reference. Please refer to the ISO and IEEE website for further details.

- **ISO 9241: Ergonomic requirements for office work with visual display terminals:** This standard provides requirements and recommendations related to attributes of hardware, software and environment that contribute to usability, and the ergonomic principles underlying them. Parts 3 to 9 contain hardware design requirements and guidance.
- **ISO 9241 parts 2-7:** It provides detailed guidance on the design of user interfaces.
- **ISO 9241-11: Guidance on Usability (1998):** This standard (which is part of the ISO 9241 series) provides the definition of usability that is used in subsequent related ergonomic standards.
- **ISO/IEC 9126: Software product evaluation - Quality characteristics and guidelines for their use (1991):** In the software engineering community the term usability has been more narrowly associated with user interface design. ISO/IEC 9126, developed separately as a software engineering standard, defined usability as one relatively independent contribution to software quality associated with the design and evaluation of the user interface and interaction.
- **ISO/IEC FDIS 9126: Software Engineering - Product quality:** It describes the same six categories of software quality that are relevant during product development: functionality, reliability, usability, efficiency, maintainability and portability. It also provides criteria for the evaluation of user interfaces and examples of metrics for effectiveness, productivity, safety and satisfaction. Specifying usability requirements and verifying that they have been achieved in a usability test is an important component of user centered design (ISO 13407).
- **ISO WD 20282: Usability of everyday products (2001).** A multi-part standard is being developed to specify the information about usability that should be provided with a consumer product, so that a purchaser can judge the ease of use of the product. It will specify a test method, the characteristics of a normal user, and how to specify the characteristics of intended users with special needs or with special skills or experience;
- **ISO 14915 (part 1-3) and IEC 61997** contain recommendations for multi-media interfaces. More specific guidance can be found for icons in ISO/IEC 11581, PDA's in ISO/IEC 18021 and cursor control in ISO/IEC 10741 to specify details of the appearance and behavior of the user interface.
- **IEC CDV TR 61997: Guidelines for the user interfaces in multimedia equipment for general purpose use (2000).** This technical report gives general principles and detailed design guidance for media selection, and for mechanical, graphical and auditory user interfaces;
- **ISO 13407: Human-centered design processes for interactive systems (1999).** This standard provides guidance on human-centered design activities throughout the life cycle of interactive computer-based systems. It is a tool for those managing design processes and provides guidance on sources of information and standards relevant to the human-centered approach. It describes human-centered design as a multi-disciplinary activity, which incorporates human factors and ergonomics knowledge and techniques with the objective of enhancing effectiveness and efficiency, improving human working conditions, and counteracting possible adverse effects of use on human health, safety and performance.

### 3.4. Other well-known HCI and usability standards

While in the previous paragraphs we have reported a long list of standards that have been formally developed focusing mainly on proprietary software, several other HCI guidances exist both for proprietary software and for separate kinds of F/OSS. Although in some cases, such guidelines have been defined by proprietary software vendors (e.g. Microsoft and Apple), nothing would prevent their full adoption to the development of F/OSS software.

- **Windows XP - Guidelines for Applications**<sup>19</sup>: These guidelines from Microsoft will help developers and designers adopt the new look and feel of the Microsoft Windows XP operating system. This collection of style elements and controls provides a base of design specifications and implementation details for using Windows XP theming in an application. These guidelines are an important resource for anyone developing applications for Windows XP.
- **Apple Human Interface Guidelines**<sup>20</sup>: This document is the primary user interface documentation for Mac OS X. It provides specific details about designing for Aqua compliance in Mac OS X version 10.4, although some of the information may apply to previous versions of Mac OS X. It explains the specific user interface components available to developers and includes extensive guidelines on how to use and implement them in applications for Mac OS.
- **NASA-STD-3000: Man-Systems Integration Standards**<sup>21</sup>. This document provides specific user information to ensure proper integration of human-system interface requirements with those of other aerospace disciplines. Video images of relevant space missions are also provided to illustrate human factors design concerns.
- **Usability.gov**<sup>22</sup> is an online HCI guideline for web applications developers.

Strictly speaking about F/OSS software, the two main projects currently active in the development of a complete desktop environment, GNOME and KDE, with a proper set of applications suitable to the needs of end-users, especially in the SO-HO segment, here are their related guidelines:

- **GNOME Usability**<sup>23</sup>: The project aims both to aid developers in their efforts to create intuitive applications, and to lead by creating designs and detailed mockups toward a cohesive and beautiful new generation of the GNOME desktop. This usability project achieves these goals through the creation of an interface guide defining and evolving the GNOME user interface, working with maintainers to find existing interaction problems through user testing, and the visual/interactive engineering of new desktop components.
- **KDE Usability Project**<sup>24</sup>: The KDE Usability Project is an initiative to apply usability principles and practices to the K Desktop Environment. Its 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.

---

19 <http://www.microsoft.com/whdc/Resources/windowsxp/default.mspx>

20 <http://developer.apple.com/documentation/UserExperience/Conceptual/OSXHIGuidelines/>

21 <http://msis.jsc.nasa.gov/>

22 <http://usability.gov/>

23 <http://developer.gnome.org/projects/gup/>

24 <http://usability.kde.org/>

### 3.5. Collaboration between usability experts and F/OSS developers

As already stated, to improve the F/OSS usability level it will be useful to involve usability experts during the F/OSS development process and, better, in the early stage of defining the software requirements specifications.

Such a recommendation is confirmed by several HCI guidelines and, in particular, in the **NASA User-Interface Guidelines**<sup>25</sup> which require that the development team and usability expert(s) meet to consider the paper prototypes and pick the best features of each design.

To help developers and usability experts collaborate with each other in an on-line manner and through the web, the website **Openusability.org**<sup>26</sup> has been created. As clearly stated on the home-page, *"...the idea behind Openusability is simple: there are many Usability Experts who want to contribute to software project and there are many Developers who want to make their software more usable, and as a consequence, more successful..."*: OpenUsability act as an electronic hub that brings Open Source Developers and Usability Experts together.

Aimed at by a similar spirit, David M. Nichols and Michael B. Twidale have launched **"The Usability of Open Source Software"** website<sup>27</sup>. The mission is clearly explained in the home-page: *"This project aims to look at the problems (and successes) of designing usable open source software. It looks at the reasons why OSS projects may pay less attention to usability than to other aspects of the software development process, and how usability issues are discussed or dismissed in different projects. Based on this analysis we aim to work towards a range of technical, educational, managerial and social mechanisms to integrate usability into open source software development."*

### 3.6. Insufficient usage of HCI guidelines and usability experts' recommendations by F/OSS developers

There are several causes related to the insufficient involvement of usability experts in the development of F/OSS projects. Here we try to briefly summarise the main ones:

- **Inertia of the F/OSS developers:** especially when acting alone or in very small projects, developers - F/OSS developers makes no exception - tend to build a "one-man-band" scenario where the individual takes decisions about every aspect of the application (interface design, system architecture, programming language and structures and usability). Even in the cases where the developer starts feeling the need to increase the number of developers, it is common to search for other developers and not for skills like HCI and interface design.
- **Usability experts are outsiders:** while the developer community is mostly aware of the dynamics of the F/OSS movement and, as such, consider the participation in a F/OSS project as a common thing, it is quite hard for professionals outside the F/OSS community to understand, to accept and to play by the (unfamiliar) rules of the F/OSS community.
- **Underestimation of usability by F/OSS developers:** F/OSS developers, especially when involved in little projects, tend to write software to solve their own needs and, as such, to release software with a cryptic user interface and/or command line. It is evident

25 [http://usability.gsfc.nasa.gov/use/Ug\\_96](http://usability.gsfc.nasa.gov/use/Ug_96)

26 <http://www.openusability.org>

27 The Usability of Open Source Software <http://people.lis.uiuc.edu/~twidale/research/ossui/>

that to broaden the adoption of F/OSS applications, the F/OSS community should continuously increase the user-base and take into accounts the needs, and the skills, of end user that are not developers and/or that are not located within the F/OSS community.

The problem of minimal or no participation of both usability experts and end-users in the F/OSS development process is also described by David M. Nichols and Michael B. Twidale<sup>28</sup>. In another paper<sup>29</sup> they review the existing evidence of the usability of F/OSS software and discuss how the characteristics of open source development influence usability. They also describe how existing HCI techniques can be used to leverage distributed networked communities, of developers and users, to address issues of usability.

It can be concluded that with the exception of major F/OSS projects like OpenOffice.org, Firefox, KDE, GNOME and few others, there is no testing of usability or insufficient testing is carried out. In such cases the presence of HCI experts at preliminary stages of development cannot guarantee good applicability.

On the other hand, it has to be said that support from large companies for F/OSS projects can result in very beneficial outcomes due to the wide set of skills and the optimal organisations that such companies have within their developer community. Some examples of this kind of involvement are that Sun Microsystems is deeply involved in the OpenOffice.org project and that Novell is involved in the development of the SUSE Linux distribution.

---

28 <http://www.cs.waikato.ac.nz/~daven/docs/oss-wp.html>

29 The Usability of Open Source Software [http://www.firstmonday.org/issues/issue8\\_1/nichols/](http://www.firstmonday.org/issues/issue8_1/nichols/)

## 4. Towards a better interaction with usability experts

---

In a distributed environment, limited and asynchronous information flow leads to a problem of low performance among developers, compared to colocated teams. In such a team, most of the communication is ideally conducted electronically (e-mails, phone, teleconferencing, emails, etc) – sometimes all team members meet in a predefined location with changing periods. For example, most KDE developers meet at least twice in a year, but some less technical and low-profile projects' developers may find it unnecessary and expensive even to meet annually..

The Internet can both be a viable and a problematic way of collaboration, since it has a dull side which limits vocal peer conversation. However F/OSS developers can benefit from distributed intelligence where there is room for usability experts. Low-budget projects can also have an opportunity to find usability people, and merge them into their projects to increase the usability profile of the resulting product.

Reitmayr and Mühling define<sup>30</sup> three basic problems that would slow down the development towards a user-engineered software:

1. The basis of usability work, a clear definition of the target users, their tasks and their requirements are often missing.
2. Missing hierarchical decision paths may pose a problem in larger projects where a usability expert needs to convince each relevant developer instead of the head of department.
3. A traditional focus on technical rather than GUI-related issues may require a major redesign of the information architecture and interaction design of software.

We can also add some items about why usability experts face problems taking an active part in F/OSS development.

1. With usability experts taking an active part in the project, the time-to-market is slowed down.
2. A developer should cope with other roles – now he is not the only decision maker.
3. The developers assume that the user base is the same as in other similar projects, so the UI requirements should be the same.
4. The necessity of a user interface developer is questioned after introduction of interface design software.

While all of the problems can be specified as critical and yet close to unsolvable in the very near future, we find that the 2<sup>nd</sup> item above (about missing hierarchical paths) deserves a more oriented focus. In most cases, HCI specialists spend considerable time to convince developers of their utility but without significant success. The F/OSS developers resist the ideas of non-developers while working on more important issues like increasing the number of feature sets of their software. Even if developers accept the ideas, they lower the priority so as to implement other important and new feature ideas before considering usability issues..

Most of the time, developers who have seen projects succeed without a considerable user centered design involvement even start questioning the roles of the usability specialists. However, as a general CRM<sup>31</sup> rule of thumb, only 5% of customers return back to the vendor for a specific product problem, and in F/OSS cases, where it is hard to talk about

---

30 Integrating Usability with Open Source Software Development: Case Studies from the Initiative OpenUsability, *tOSSad OSS 2006 Workshop proceedings*, pp. 65

31 Customer Relationships Management

vendors and customer satisfaction, the feedback is often disregarded, if not ignored.

While little or no, knowledge exists among F/OSS projects and developers, it should be noted that three (Kuroo, Mandvd, Amarok) of top four KDE applications listed on the KDE applications web site<sup>32</sup> with the highest grades has neither usability evaluators or developer support (Kuroo) involvement, or has received usability reports (Amarok) or has usability as the first aim of the project (K3b). This clearly shows the importance of user interface amendments in a F/OSS project.

#### 4.1. Collaborating with usability experts

The first rule of thumb to collaborate with usability experts is to find them. Without a proper search, no usability expert will ever know about the project and achievements. There are mailing lists and discussion forums that such experts subscribe to. The following table gives an overview of most known mailing lists at the time of writing.

Table 1: Usability related mailing lists

Mailing list name	Subscription address
KDE usability	<a href="https://mail.kde.org/mailman/listinfo/kde-usability">https://mail.kde.org/mailman/listinfo/kde-usability</a>
GNOME usability	<a href="http://mail.gnome.org/mailman/listinfo/usability">http://mail.gnome.org/mailman/listinfo/usability</a>
ACM SIGCHI human factor aspects of the web	<a href="http://www.sigchi.org/web/">http://www.sigchi.org/web/</a>
Interaction design	<a href="http://lists.interactiondesigners.com/listinfo.cgi/discuss-interactiondesigners.com">http://lists.interactiondesigners.com/listinfo.cgi/discuss-interactiondesigners.com</a>
Usability matters	<a href="http://groups.yahoo.com/group/usabilitymatters">http://groups.yahoo.com/group/usabilitymatters</a>

Having a profound knowledge on usability and related tasks takes some time. The following table contains a list of usability related resources which serves as a starting point.

Table 2: Usability related resources

Resource name	Web address
Alertbox, Jacob Nielsen's bi-weekly column for usability	<a href="http://www.useit.com/alertbox/">http://www.useit.com/alertbox/</a>
Usability Professionals Association Resources	<a href="http://www.upassoc.org/usability_resources/">http://www.upassoc.org/usability_resources/</a>
STC Usability SIG	<a href="http://www.stcsig.org/usability/topics/index.html">http://www.stcsig.org/usability/topics/index.html</a>
Resources by Darlene Fichter	<a href="http://library2.usask.ca/~fichter/usability">http://library2.usask.ca/~fichter/usability</a>
Usability in Wikipedia	<a href="http://en.wikipedia.org/wiki/Usability">http://en.wikipedia.org/wiki/Usability</a>
OpenUsability	<a href="http://www.openusability.org">http://www.openusability.org</a>

A good practice to find a relevant usability experts includes the following steps:

1. Subscribe to an appropriate mailing list.

<sup>32</sup> Listed in <http://www.kde-apps.org>

2. Choose a relevant subject for your “call for help” e-mail.
3. Describe your project, aims and vision. An convincing vision might stress the usability orientation of your project.
4. Explain how your project can benefit from a (heuristic<sup>33</sup>) usability evaluation
5. Mention how a usability expert can benefit from your project. For example, those experts can benefit from the practice and add another use case to their background.

Candidate experts may or may not know how to collaborate in a geographically dispersed and asynchronous manner, so a first mentoring about how things work in a F/OSS world can be time saving. This way, details about relationships within the team, differences in time planning and project planning, how to handle future features, bug fixing and team management can be thoroughly learned and absorbed. These items need a sincere understanding by all the volunteers who are new to the project.

The time at which a usability expert becomes involved in a F/OSS project is critical. Early involvement offers a better chance of a strong influence on the product user interface, since the acceptability of the interface designer will likely be higher within the project. In this way, a paper prototyping is possible, allowing continuous amendment cycles during the product design stage. Moreover, the usability expert will not only develop user requirements and develop user profiles, but he will also be able to consistently fix and/or offer usability bugs found within the product at an early stage. The table below shows different involvement stages of an usability expert in a F/OSS project and their outcomes.

*Table: Different involvement stages and their outcomes*

	<b>Early involvement</b>	<b>Halfway involvement</b>	<b>Late involvement</b>
Influence on the product user interface (UI)	High	Moderate	Less likely
Acceptance of the expert	High	Moderate	Moderate
Applicability of paper prototyping	For all UI elements	For the upcoming UI elements	Almost none
The focus point of the usability expert	Developing user requirements	Amendment of user interface	Crucial usability bug fixing

## 4.2. Communication of developers and usability experts

Usability experts have traditionally worked on-site, with test subjects, clients, developers and project managers. Now they face the challenge of asynchronous communication. Many tools have been developed to remedy the situation where few developers work in the same ( physical ) place.

Daily meetings may not be possible in a F/OSS project, and the unavailability of developers may lead to frustration among usability experts. Since there is no well-defined interaction template between a usability expert and a software developer, this may yield unwanted results. If a meeting is to be held, whether via teleconferencing or an on-line system (i.e. IRC or chat) an agenda should be fixed beforehand to increase performance.

<sup>33</sup> Heuristic evaluation can be stated as an expert usability testing method that helps to find usability problems in the user interface design. it is a quick, cheap and easy evaluation for user interfaces which are commonly executed by experts.

Many, if not all, projects use a repository control system (i.e. CVS<sup>34</sup> or SVN<sup>35</sup>). Using CVS or SVN is a way for coders and developers to submit patches, fix bugs and track/freeze various versions of code. With CVS or SVN, every change of a file is logged and many developers can work off the same code base at the same time. Another alternative is a file tracking system, with where developers can upload files they work on, and set a comment for each file.

The problem with CVS or SVN which can effect an usability expert comes in two flavors:

1. Such repository control systems need a client to enable access . These clients often present an unintuitive way for usability experts to reach files/folders in a given repository.
2. It needs some time for the nontechnical user to become used to CVS or SVN because of newly introduced tasks like checking in, checking out, committing, etc.

A F/OSS project utilises more of CVS and/or SVN for information sharing. Mailing lists, forums, trouble tickets have been traditionally used to ease communication. It has always been a challenge to teach usability experts to use these applications, services and concepts. The project manager should be flexible enough to change and/or modify the aforementioned items in order to build a healthy environment which can be exploited by usability experts.

#### **4.2.1. Bugzilla: friend or foe?**

Bugzilla is a renowned tool for on-line software development, which can be used to track errors, wish lists and defects in a project. This application has traditionally been the de-facto standard for many projects to track bugs and wishlists. Bugzilla's extensive features and complicated user interface is accepted by many developers, but it takes extra time to get used to for non-techies. Consequently, those who do not know how to use Bugzilla can easily get frustrated with the high number of questions to be answered and the different fields to be filled. Graphics designers, documentation writers and translators may find it hard and eventually stop using the Bugzilla interface. Inexperienced users may submit bugs which are related to usability but choose a wrong component, or which are not related to usability but choose the "usability" component. Such problems should be handled politely by the developers first, notifying the usability expert (who, initially, has insufficient skills with Bugzilla).

If the usability expert has no idea what Bugzilla is, a choice by the project lead developer can be given so that either

1. The corresponding usability expert uses another tracking tool which is more user friendly, or
2. The usability issues can be handled without a tracker.

Before the involvement of usability expert into the project starts, the above mentioned matters should be taken into consideration and necessary steps should be taken.

However, we were unable to evaluate any bug reporting tools which have usability extensions like posting videos or pictures with bullet or note adding features. We believe that this issue deserves research and the development of a corresponding tool, enabling a framework where usability experts can easily post their bug reports.

It should also be mentioned that a bug reporting tool is a means to report bugs and wishlists, and not a place for discussion. Any attempt at discussion within the tool will

34 Concurrent Versions System

35 For more information about SVN (subversion), see <http://subversion.tigris.org>

cause annoyance, alienating the other developers on the mailing lists. A dedicated discussion system separate to the bug reporting tool is required.

#### 4.2.2. Collaboration on mailing lists

A mailing list is the heart of a F/OSS project: it is the arena where (distributed) developers share opinions, accept or reject other's ideas and develop the future of the software. A usability expert should be in touch with other developers necessarily, as new ideas for interaction are identified and accepted by the project leader(s). Just like other tools, mailing lists pose a cold interface which do not carry emotions. This is a contradiction to the fact that usability experts have traditionally worked with users and customers interactively to build usable interfaces. While the nature of the internet has brought up many challenges in this domain like remote usability.

It is likely that the usability experts may not have been active members of a mailing list before. This may make it hard to exchange information on the mailing list. Before faced with such a problem, it's wiser to identify the collaboration rules for all developers and users.

For a fruitful discussion, the following items should be kept in mind.

- Every mailing list should have information on how to unsubscribe.
- The mailing list manager can periodically send an e-mail to the list, stating the purpose and aims, together with who is involved in which major parts of the project.

A mailing list also hosts the future announcements on a specific (sub)application of the project. Before such announcements, the developers should make sure to include the opinions of the usability expert, since this person may also show a flag if i.e an interface component do not comply to the standards, or it is unintuitive, or hard and confusing to finish a task in a given period of time.

#### 4.2.3. Communication via project web page

As a general rule of thumb, a project's web page, its navigation, attractiveness, ease of use, frequency of update, maintainability and other features often define the success or failure of a project in our well-connected world. People would like to see an active web page, stating the project's main goals, objectives, road maps and a guide to developers, presented in a clear manner. Without these items, a project, especially with a lack of funding barely connects with the outside world.

Below, we have stated some communication tips which will help usability experts to integrate themselves into the project, and provide their ideas and criticisms in order to build a usability-centric application project.

**Adding a to-do list:** it is astonishing to see how few projects actually have a basic to-do list which will guide a newcomer through the project. A usability expert would like to see which parts of the software need improving, and act accordingly. A note on the project web containing the list of to-do's will also drive other potential contributors to the project.

**Usability junior jobs<sup>36</sup>:** Many major F/OSS projects have introduced a facility on their web page lately to attract newcomers and non-technical people who do not have a high degree of expertise, but would like to join and feel themselves an important part of the project itself. Junior jobs started to become a major point of entrance to distributed F/OSS development, and usability junior job listings can be an effective way to attract usability

---

36 KDE Junior Jobs on <http://tinyurl.com/preview.php?num=ab7l3>, accessed 28 July 2006.

experts. In order to exploit this, project maintainers must be sure to list the usability tasks clearly, and assign roles if the particular requirement needs more than one person. Another useful method is to give experts a starting point, (i.e. adding a list of sub-tasks, preparing a usability report template, linking to other projects with same ambition or framework, etc.).

**Collaboration with [openuability.org](http://openuability.org):** [Openuability.org](http://openuability.org) has an aim to gather usability specialists and F/OSS projects under one umbrella. Register your project under [openuability.org](http://openuability.org) and search for a usability expert who can give you support with one of the items mentioned in chapter 4.2.4: *Some common Usability Methods*.

#### 4.2.4. Some common usability methods

A F/OSS project benefits not only from the visionary perspective of a usability expert, but also from his usability studies and initiations. Some of these methods and techniques are stated below. We are listing these methods to ensure that developers become aware of what kind of ways are exploited in making a product more usable and acceptable to the end user. Note that not all these studies can be relevant to all F/OSS projects, and it is rarely useful to refer to all of them.

- **Awareness building:** The usability expert can inform the developers about the usability and interaction design basics by forwarding e-mails, giving short seminars, preparing peer to peer talks, etc. This way, the developers will be able to take necessary precautions during the user interface design process.
- **User focused requirements analysis:** Before developing a product, the basic requirements should be determined and documented. In this simple document, target audience, features and task completion benchmark (i.e. a task should be completed within 5 minute) can be identified. In this stage, a participatory approach should be carried out to involve users in determining the requirements through previous studies, interviews, surveys and focus groups.
- **Building user profiles:** Building the general user profiles require a set of considerations and answers to questions like<sup>37</sup>:
  - Who are the users, what do they know, and what can they learn?
  - What do users want or need to do?
  - What is the general background of the users?

Answers to these questions help the developer build a user profile and modify the design of the application necessarily. Other types of questions and/or remarks include the necessity of training, degree of accomplishment of a typical user for a typical task, the suitability of error messages and the accessibility concerns of the particular application.

- **On-line and off-line surveys:** Subjective surveys with open ended and closed questions help identify the user's expectations of the software and lead to an understanding of their cognition.
- **Usability testing** includes the testing of a set of potential product users with a set of tasks. Observers watch the test subjects using the software. Usually, a camera is used for recording to help future inspections. Generally, the accuracy of the test subjects, time to finish a specific task, the training needs, recovery from errors, emotional responses and mental burden are calculated and then analysed. While usability testing is usually a time consuming task, its outputs are very valuable since subjects from the real world are included in the validation phase. KDE and GNOME projects have benefited from various usability tests and amended the interaction

---

37 Usability subject on Wikipedia, <http://en.wikipedia.org/wiki/Usability>

structure and visual appearance of the corresponding applications. One example of usability testing is eye tracking. A camera focuses on one or both eyes and records their movement as the viewer looks at some kind of stimulus. Most modern eye trackers use contrast to locate the center of the pupil and use infrared light to create a corneal reflection. Scan paths are useful for analysing cognitive intent, interest, and salience<sup>38</sup>.

*Figure 2: Typical eye tracking system using head mount*



- **Guideline development:** The developers need reliable guidelines to ensure consistency; usability experts can develop guidelines to build a basis for the interface basics. Even a simple guideline with details like the use of colors, terminology, design style, icon guides and use of keyboard and mouse is a good starting candidate.

The methods described above help in making a product usable.

---

38 [http://en.wikipedia.org/wiki/Eye\\_tracking](http://en.wikipedia.org/wiki/Eye_tracking)

## 5. Summary

---

In this report, we focused on the involvement and the adoption of usability experts into F/OSS projects. We analysed the problems and how to overcome them in a straightforward manner. One way to get usability experts into F/OSS projects is to make developers more aware of basic usability principles. The outcome of this action is:

1. Developers will be able to evaluate their own projects, which in turn yields better quality and usable software.
2. There will be a mutual understanding of two groups (i.e. developers and interaction designers).
3. A usability expert will show the user's perspectives and focus on user profiles, finely defining the target groups.

Some can argue that a well-defined F/OSS project methodology which includes usability experts will be worthless since it undermines the philosophy of distributed project management. This has proven to be true in some projects, where the project manager consistently focuses on features more than usability. However, starting from year 2000, this phenomenon has greatly changed from an ignorance of usability aspects to a total respect towards a user centered design.

We see the absence of key components like usability reports, usability laboratories, usability experts and companies specialising in usability for F/OSS, to be taken in appropriate consideration during all of the development process<sup>39</sup>.

The way out from the current situation is a proper combination of the following factors:

- Creating a pool of knowledge in the usability field that could interact with F/OSS developers, through the web;
- Creating web resources for community usability experts and F/OSS developers like on-line books, references and guidelines about usability and usability HOW-TOs.

For a cardinal change of the current status, not only it is necessary to take into account usability standards requirements, but it is critical to adopt them as early as possible in the development process. Such an approach, in effect, can be seen like an adaptation and/or a simplification of the directives specified by current proprietary standards provided by external organizations.

---

39 <http://www.newsforge.com/article.pl?sid=04/07/07/1640244>