OpenUsability.org
Usability and Open Source Software

Open Source Software (OSS) is not notable for being usable for the average user. Some (partially self-acclaimed) 'experts' see this as being due to a lack of interest by the non-commercial developers in ergonomics and usability. Our experiences with several open source projects prove a less stereotypical picture.[1]

In this article, we show some structural settings, or general conditions, of OSS projects and outline their effects on usability in comparison to commercial software development. We describe the challenges of realising and establishing usability in the open source realm for the future. The focus is on two factors: the integration of usability into the daily practice of OSS development, and the availability of usability resources (experts, knowledge, time, availability). If this can be achieved, OSS has an excellent chance of making usability a key market advantage.

Introduction

The success and spread of OSS in recent years is based mainly on server and backend applications. Its success is still limited on the desktop. Beacons, like the decision of the cities of Munich and Vienna to migrate to Linux, might lead to the conclusion that it is only a matter of time for the OSS success story to be continued on the desktop.

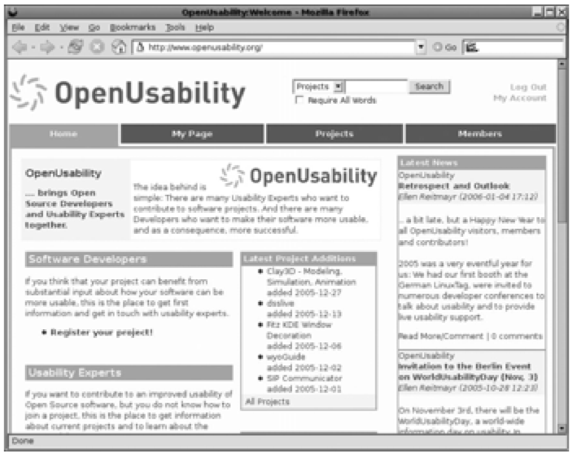
For the OSS desktop to compete with commercial software, OSS must be usable, not only for the 'geeks', but also for the 'average user'. But how usable and user-friendly is OSS? There are currently few studies.

Berlin-based user experience consultancy Relevantive undertook a study in Spring 2003 about Linux in a technically administered environment (Muehlig et al. 2003); this was one of the first studies to provide data about this issue. Since then, hardly any further studies have been done. Also, little is known about the role of usability in OSS projects. Nichols and Twidale (2002) were pioneers in this field, yet their conclusions are drawn mainly from structural factors of Open Source development, not from concrete empirical data.

In this article, we discuss some of the factors that sustain or hinder the development of usable Open Source Software and relate our experiences of several projects, including the KDE desktop. In addition, we want to show that Open Source is a wonderful realm for applying and enhancing usability methods and to promote the use of usability to both developers and a broader audience.

Developing for the community

One of the most frequent arguments about why OSS performs poorly with respect to usability (a charge made, in many cases, without concrete evidence) is the traditional orientation of developers towards the community, that is, other hackers (Nielsen 2004). The community judges the quality of a piece of software so it is, to some degree, written to their requirements. Other projects are developed to meet an individual programmer's requirements. So can you demand that this software is usable for non-geek users? And


OpenUsability:Welcome - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.openusability.org/
Go
OpenUsability
Projects
Search
Require All Words
Log Out
My Account
Home
My Page
Projects
Members
Done

OpenUsability

... brings Open Source Developers and Usability Experts together.

The idea behind is simple: There are many Usability Experts who want to contribute to software projects. And there are many Developers who want to make their software more usable, and as a consequence, more successful.

Software Developers

If you think that your project can benefit from substantial input about how your software can be more usable, this is the place to get first information and get in touch with usability experts.

- Register your project!

Usability Experts

If you want to contribute to an improved usability of Open Source software, but you do not know how to join a project, this is the place to get information about current projects and to learn about the

Latest Project Additions

- Clay3D - Modeling, Simulation, Animation added 2005-12-27
- dsslive added 2005-12-13
- Fitz KDE Window Decoration added 2005-12-06
- wyoGuide added 2005-12-02
- SIP Communicator added 2005-12-01

All Projects

Latest News

OpenUsability
Retrospect and Outlook
Ellen Reitmayr (2006-01-04 17:12)

... a bit late, but a Happy New Year to all OpenUsability visitors, members and contributors!

2005 was a very eventful year for us: We had our first booth at the German LinuxTag, were invited to numerous developer conferences to talk about usability and to provide live usability support.

Read More/Comment | 0 comments

OpenUsability
Invitation to the Berlin Event on WorldUsabilityDay (Nov, 3)
Ellen Reitmayr (2005-10-28 12:23)

On November 3rd, there will be the WorldUsabilityDay, a world-wide information day on usability in

that users who cannot intuitively use such software should simply choose another?

This argument is quite plausible, but it is no longer valid if you want 'your' software to be considered seriously in businesses, public administrations, or converting end-users to OSS.

While, traditionally, the success of OSS was determined more by performance and functionality than ease-of-use and aesthetic appeal, it is also true that hackers are normally proud if their software gets widely used. And, as the acceptance and the demand for OSS from a larger audience increases, we are finding that developers and projects more and more acknowledge, and even welcome, the need to make their software more usable for a wider range of users.

In principle, usability relates to certain users in certain contexts. In general, a clear description of the target user groups is lacking from OSS development – commonly the users are seen vaguely to be 'everyone'. Even if information about the users were available, the desire to make software usable for users other than hackers consequently leads to a significant conflict.

Making software usable for one user group is a feasible challenge, but for 'everyone' tends to be difficult if not impossible. This leads necessarily to prioritisation, which again is problematical for political reasons. For example, most OSS has a huge range of functionality, which matches the needs and expectations of the developers (who can cope with the complexity) but, for 'average users', the amount of functionality renders the software unusable.

Take a real-world example: the email client Kmail. It has a setting that defines a mail folder as containing a mailing list. One developer designed this function, programmed it, and integrated it, but the number of users who deal with mailing lists outside the hacker community is rather limited. Nevertheless, this function is displayed on the same level as folder names and icons.

For the non-geek user, the amount of functionality makes the usage difficult because he cannot immediately distinguish between essential and optional functions (for his purpose).

Does this lead to the conclusion that the program should only offer those functions that are relevant for average users? Which functions will be removed? What will be the response of those developers who wrote these functions? What will be the response of the users who chose the software exactly because it had those functions?

These conflicts became prominent when we worked on a redesign of the folder menu dialogs of KMail, to make them more usable for a wider audience. Removing or hiding exotic functions would reduce the cognitive load and visual clutter, and would likely support a quick perception of essential functions. On the other hand, some developers did not like these 'improvements' and questioned the competence of the usability experts.

Whether a project decides to prioritise the average user and increase compatibility beyond the hacker community will surely become more prominent in the near future. There may be space for compromise, but this conflict is not easy to bridge.

Usability is trivial?

In a highly disputed article, the renowned OSS evangelist Eric S Raymond curses the poor usability of the CUPS configuration system (a Unix printing interface). Raymond attacks the developers for their design negligence and concludes that they only needed Aunt Tilly in mind to design the software so that it was usable for her (and thus for everyone else too): Usability is trivial.

This assumption is well established in OSS, as well as in commercial software development. In fact, a fundamental misconception about usability is obvious: How do I know what Aunt Tilly needs? Have I observed her using the software? Do I know why she does what she does? Do I know which terms and concepts she understands? Probably not.

Instead, an image of a fictitious aunt and her usage environment is made up to represent the lower end of all users. Gruber (2004) is absolutely right when he points to the disdain of 'dumb users' that is implicit in this attitude. OSS seems to be more predisposed for such a stance, since its success is not necessarily dependent on a market, unlike commercial software.

If usability were so trivial that you only have to think about the right user, we would live in a world in which software disappears because it matches our needs so well that we wouldn't notice it anymore. Instead, usability in most cases is quite a laborious process. The software is tailored to the user so that it is intuitively, successfully, and efficiently usable in an enjoyable way.

In classic software development, this aim is ideally achieved by collecting and analysing data about the target users (the requirements) and then by developing the software, step by step, while checking it against the reality, the user, using prototypes. This procedure is complex and expensive, requires usability specialists and, of course, must be backed by the project management. If not, it is merely a lip service. OSS projects can decide for themselves if their software should be suitable for average users. If the developers make this decision, they must equally set up workflows and resources to achieve this goal.

Usability as a bazaar?

Collaboration in OSS projects was compared, by Raymond in his famous article 'The Cathedral and the Bazaar', with the mechanisms of a bazaar. OSS development is characterised by open communication structures: it is easy to get in contact with the developers and to provide feedback. Consequently, OSS should have the ideal conditions to achieve usable software through interaction with users. Can usability contributions work similarly? Or are there significant differences between usability contributions and bug fixes?

Maintainers of OSS projects can judge very well what is good code and what is not, even if they don't know the contributor. Bugs are usually objectively reproducible and are identifiable as either really a bug or not.

This is different for usability contributions. Let's say someone sends in a description of a usage problem and a solution to the project maintainers. How do the maintainers know that it is a real problem? How do they know that the suggestions really do solve the problem and for which users? From this, it is obvious that usability contributions cannot be be handled in the same way as code.

Aside from this, consistency or conformity to guidelines can easily be verified. Still, guidelines cannot sufficiently describe how to design a usable interface.

Without this basic foundation, usability becomes mere speculation and the aim of getting a more usable application for non-geek users cannot be achieved. These forums more often present personal opinions of users rather than applied usability knowledge. As Nielsen has repeatedly stated, 'users are not designers' and 'designers are not users' (1994).

It seems unlikely that the 'wonder of the bazaar', where everybody speaks at the same time and, by way of magic, the right result comes out, does work for the field of usability. On the contrary, in the midst of many voices (and many opinions), the maintainers have difficulty trusting such opinions, and filtering noise to make a decision. Often a decision is never made and the software is left untouched and unimproved. This will only change if discussions are based on facts and mere opinions are better filtered.

Important parts of the bazaar are the communication channels on which the projects mostly rely: mailing lists, IRC (a kind of chat room), and bug tracking. These channels are well suited for technically savvy users, but are very difficult for non-geek users. In fact, they work as a filter distorting the representativeness of user feedback. But assuming that they were easy to access and to use, they still would be of little help for OSS projects. Imagine if thousands or millions of users report their problems, wishes, and solutions to the programmers, there would be no time left for coding. The problem of 'who is right' and 'whom shall I believe/trust' is still the same.

Missing resources

From what we have described so far, it becomes clear that OSS projects have a fundamental problem: they lack usability resources that help achieve better usable software for non-geek users. The community has traditionally consisted of programmers, while usability experts are practically absent. Even in large desktop projects, like KDE, there is only a handful of members with strong usability skills.

There are exceptions where companies try to support OSS projects with their own experts (for example, OpenOffice.org and Sun Microsystems), but they are commercially motivated and do not always get the wide acceptance by the community. Also, the conflict of 'developing for the community' versus 'developing for the average (or marketable) user' gets even more emphasised.

Speed is the key

So far, we have mainly described the difficulties and handicaps that OSS faces with respect to becoming usable software. OSS has, however, a huge advantage over commercial software: the principle of 'release early and often'.

Professional usability engineering is effective when the software is presented to the user early in its development. The earlier such tests can be done, the easier and cheaper are changes and adaptations. This is where prototyping comes into play. Unfortunately, in commercial reality, it is neither done as often as expected, nor as early as needed.

For OSS, however, a constant publication of incremental 'prototypes' is simply part of the open source process: a fantastic situation for every usability engineer! With frequent releases, it is possible to integrate improvements step by step. Classic software development has a much more rigid framework which focuses on major releases and does not allow for many iterative usability changes. Here, OSS has a huge advantage – a killer feature – that can lead to better and more usable software but it requires usability resources.

Open Usability

In OSS development, usability experts are very rare, and procedures to integrate them are missing. On the part of potential usability contributors, knowledge about the peculiarities of OSS development is missing – it is unclear what is expected and how to interact with the developers. On the part of the developers, it is unclear what they can expect from the usability experts, what is needed by them, and whether the contributions from the so-called usability experts can be trusted.

Changing this situation is one of the main motivations of the OpenUsability project. The idea was born at the KDE developers conference 2003 in Nove Hrady (Czech Republic) to which OpenUsability founders Jan Muehlig and Jutta Horstmann were invited. They discovered that interest in usability was very high but knowledge about usability and how to achieve it were lacking.

The result is a portal that aims to facilitate the interaction between usability contributors and developers. Projects can present themselves and communicate their wish to incorporate usability and improve their software – this is crucial to avoid misunderstandings and frustration because not all developers and projects are sold on usability. Likewise usability experts can get in direct contact with project maintainers or representatives and get a clear picture of what is needed.

Currently, the portal is being redesigned as a result of our experiences with many of the participating projects. We are looking forward to providing effective means and workflows for collaboration, as well as documentation, HowTos, and a general interface between OSS projects and usability resources.

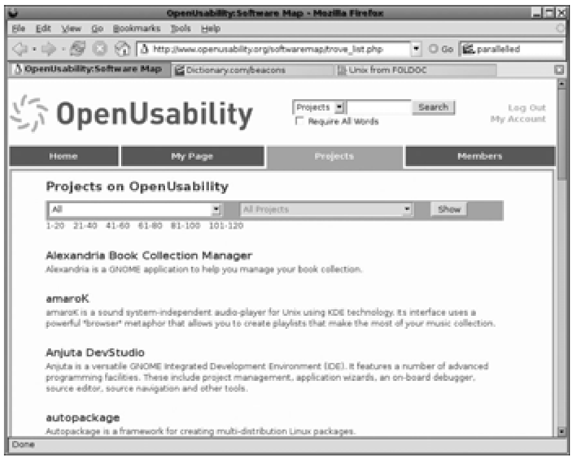
Trust

One thing we have definitively learned during the last few years is the importance of trust. Typical open source projects rely on remote communication tools like mailing lists, IRC, bug tracking systems and code repositories.

Collaboration does not depend on seeing or sitting next to each other because code can largely speak for itself. Physical meetings are mostly not crucial for day-to-day development. But, as we said, usability input is different to bug reports and, in a world where many feel inclined to provide their 'usability expertise', it is difficult for developers to know who to believe and follow – or not. Therefore, personal relationships between usability engineers and developers are very important.

The developers are ultimately those who decide which changes do and do not get committed. If the developers do not trust the usability input, the changes will never get made. There is no executive power to force usability changes on an OSS project – rightfully so because many of these projects are very personal to those who maintain them.

While this may sound counterproductive at first, it is ultimately very fruitful. Established usability engineer–developer relationships have a high success rate of committed usability improvements due to the willingness and trust that the developer has for the usability engineer. The 'review' period of the usability feedback is greatly shortened if not eliminated because they have proved to be a trusted source.



Projects on OpenUsability

Alexandria Book Collection Manager
Alexandria is a GNOME application to help you manage your book collection.

amaroK
amaroK is a sound system-independent audio-player for Unix using KDE technology. Its interface uses a powerful "browser" metaphor that allows you to create playlists that make the most of your music collection.

Anjuta DevStudio
Anjuta is a versatile GNOME Integrated Development Environment (IDE). It features a number of advanced programming facilities. These include project management, application wizards, an on-board debugger, source editor, source navigation and other tools.

autopackage
Autopackage is a framework for creating multi-distribution Linux packages.

Invitation

Open Source Software has specific disadvantages with respect to usability. This includes the traditional focus on code, the strong preference of functionality over accessibility, and a lack of good usability amplifies this. The current poor attraction and integration of usability contributors and the non-availability of usability resources are fuelling this even more.

But many projects are willing to become usable to non-geeks and to become the first choice of end-users. The interest in usability by the KDE and Gnome projects, and the many projects that participate in OpenUsability and in other efforts like the FLOSS-Usability group, is really promising. If OSS incorporates usability as a significant part of its development process, it can make full use of its advantages, including its 'rapid prototyping' framework. In which case, OSS may one day be notable for its usability.

In addition, Open Source usability can be an extremely rewarding field for any usability professional who wants to improve software for the sake of applying and sharing skills and knowledge. We invite everyone to contribute to a project and to take advantage of the open environment to provide free, usable software to the world (as well as sharpening your skills in the process). And it is a unique opportunity to help improve the free and open software you can use everyday.

References

Gruber, J. (2004), *Ronco Spray-on Usability*, http://daringfireball.net/2004/04/spray_on_usability, accessed 04 Feb 2006.

Muehlig, J., Horstmann, J., Brucherseifer, E. and Ackermann, R. (2003), *Linux Usability Study*. http://www.relevantive.de/Linux-Usabilitystudie.html, accessed 04 Feb 2006.

Nichols, D. M. and Twidale, M. B. (2002), *Usability and Open Source Software*. Working Paper 02/10, Department of Computer Science, University of Waikato, New Zealand, http://www.cs.waikato.ac.nz/~daven/docs/oss-wp.html, accessed 04 Feb 2006.

Nielsen, J. (2004), Developer Spotlight: Jakob Nielsen, Builder AU, http://www.builderau.com.au/webdev/0,39024680,39130602,00.htm, accessed 04 Feb 2006.

Nielsen, J. (1994), *Usability Engineering*, Morgan Kaufmann, 12–13.

Raymond, E. S. (1998), *The Cathedral and the Bazaar*, First Monday **3**(3). http://www.firstmonday.org/issues/issue3_3/raymond/, accessed 04 Feb 2006.

Raymond, E. S. (2004), *The Luxury of Ignorance: An Open-Source Horror Story*, http://www.catb.org/~esr/writings/cups-horror.html, accessed 04 Feb 2006.

1 See *Interfaces* 60 for a brief introduction to Open Source Software.

Jan Meuhlig
jan.muehlig@relevantive.de
Celeste Lyn Paul
celeste@kde.org

Jan Muehlig is CEO of the Berlin-based user experience consultancy Relevantive, and president of the non-profit foundation OpenUsability.

Celeste Lyn Paul is senior interaction designer for User-centered Design, Inc. of Ashburn, Virginia, and one of the usability maintainers for the KDE project.

Workshop Report

Peter J Wild and Claudia Eckert

International Workshop on Understanding Designers'05 Aix-En-Provence 17–18 October 2005

About 40 researchers from around the world gathered in Aix-en-Provence to report on efforts to study designers in a variety of domains using a variety of empirical methods. The researchers came from a range of academic backgrounds and domains of interest, but an important feature of the workshop was in bringing researchers and educators together; the workshop format allowed for long and lively discussions.

The major emphasis of the research studies was on the conceptual phase of design, and studied this using experiments on design students. A smaller number of papers, mainly from an engineering design background, reported on empirical studies of large-scale engineering projects or interviews with experienced designers spanning the entire design process. These papers had a wider scope then the three main themes that John Gero identified in his summary:

1. Creativity and Design (both explicit and implicit),
2. Externalisation through sketching, drawing, etc, and
3. The connections between teaching and learning.

To us this workshop was once again a reminder that truly understanding designers, if not design, embraces understanding the differences as well as what is common between different design domains. Some of the observable differences include the tension between a novel or creative solution and a well-engineered product; validation of findings; accounting for the strengths and shortcomings of your chosen research method. This sort of event makes it possible to discuss these issues. However, we concur with John that the study of design processes has just begun.

Information about the programme can be found at: http://www.arch.usyd.edu.au/kcdc/conferences/sd05/

Peter J Wild
Innovative Manufacturing Research Centre
University of Bath
peter.j.wild@btopenworld.com

Claudia Eckert
Engineering Design Centre
University of Cambridge