# Open Source Platforms and Disadvantaged Communities. *

Collin Anderson
University of North Dakota Kiwi Project
Grand Forks, North Dakota
collina@gmail.com

## ABSTRACT

During the Winter of 2007, the University of North Dakota Kiwi Project was established as an effort to put free computers into disadvantaged households with an open source platform and engage students on social issues pertaining to their profession. Two years later, the effort has introduced hundreds of individuals to free software and improved access to the technological revolution that is too often denied to low income households. Despite statistical successes, there remains a lingering feeling that the environment chosen has inhibited the project's potential. While the adoption of Linux in such projects is financial and legally free, implementation has encountered numerous hidden burdens and social impediments which have greatly changed our original equation.

In its current state, the wide-scale adoption of Linux would reverse a decade's worth of standardization and needlessly increase the learning curve of computer education for vulnerable populations. In light of this, instead of solely heralding triumphs, it is necessary at such ventures to address the points of weakness endemic in the open source model and how they pertain to computer science education.

## Categories and Subject Descriptors

K.4.2 [**Social Issues**]

## General Terms

Design, Economics, Human Factors, Standardization

## Keywords

Linux, Computer Access, Interface Design

## 1. BEING DIFFERENT

It's important to concede the obvious; Linux will always have the significant flaw of being "not-Windows." Any weaknesses in the community and its model are only one part of the explanation of why Linux has not reached a critical mass of users in the general populations. Even if Microsoft were to immediately quit every semblance of anti-competitive behavior, its platform would dominate the landscape for at least another decade. Most large enterprises are too heavily invested, in terms of education as well as money, in such platforms to justify massive upheaval. Furthermore, it is necessary to recognize that Microsoft products and concepts dominate the popular lexicon of computing. To most, Microsoft Word is the entire concept of word processing and will remain to be for some time. Similar popular, monogamous relationships exist between concept and implementation for many Redmond products, especially Excel and Powerpoint. While there have been substantive victories in forcing governments to adopt open standards and open software, experience in specific proprietary programs is still necessary for employment beyond manual labor.

Defining a principle of transparency as proximity to the current design and usage norms (today, Microsoft norms), then a decrease in familiarity is an increase in end-user burden. In practice this means: if the Kiwi Project introduces "not-Office," then at some point the concept of file types, fonts and conversion will come up. This is a decrease in transparency, which means more to learn and more anxiety. In being "not-Windows," concerned developers should therefore understand that every step away from Windows norms increases the stress of converting to a different platform. People are remarkably adept at finding subtle changes in what they are use to. Deviation from Microsoft interface fundamentals, such as toolbar placement and function names, should be balanced against the real possibility of scaring away less intrepid users. With this in mind, open-source advocates should explore development issues as an additional, alternative cause for free software's shortcomings as well as a significant constraint to the aspirations of the effort.

Firefox, the open source movement's star pupil, is likely an exception that proves the point. It seems probable that the advertising campaign to label Firefox as the 'safest way to explore the Internet' has been more responsible for its success than external application features. To its credit Firefox handles complex and wide-ranging types of media, yet its most important aspect is that it is a convenient, unimposing shell for content. In spite of this level of transparency, Microsoft branding has maintained Internet Explorer's position as the default browser, with many developers creating pages specifically for its poor implementation of web standards.

Considering the demands of the current workforce, to promote anything other than popular and uniform norms is a

*Growing the HFOSS Community, Humanitarian Free and Open Source Software (HFOSS) Symposium March 10th 2010 

gamble with the wrong group of people. Often, open-source advocates fall back on precept that individuals should learn general concepts instead of the placement of pertinent buttons. While admirable, this approach is much more instructionally intensive, a luxury for those trying to learn job skills quickly. Computer education runs contrary to the common analogy of learning to drive a car[1] in that software applications are orders of magnitude more complex and ambiguous in interface than any automobile. People have had a hundred years to move past fear and awe of a car, as opposed to one or two generations with computers.

Standardization is not simply a product of openness and compatibility. Whereas, Microsoft presents a uniform platform of the Windows Desktop, Office and Internet Explorer and Apple offers OS X, iWork and Safari, Linux is expressed through a multifaceted combination of application which are often completely distinct in appearance, name, configuration and functionality from each other. Firefox's shortcomings in adoption, despite its enlightened development, highlight the problem of bundling and uniformity in distribution that is endemic exclusively to F/OSS. Unlike its proprietary competitors, Linux has yet to, and is unlikely to ever, settle on a singular user interface, much less a browser.

## 2. BEING IDEOLOGICAL

More visibly, the issue of developer hubris and ideological entrenchment has been a reoccurring impediment to delivering a polished product to end-users. The open source community has embraced the notion that software development is meritocratic. In reality, this has created self-serving libertarian tendencies and provincialism. The open source model breeds both a selective commitment to those features that are pioneering or popular, as well as a misaligned expectation of self-sufficiency. By acting in this way such developers have already disenfranchised users who are more concerned with learning basic productivity skills than compilers, codecs, toolkits and versioning systems. There is a frightening scarcity of developers with the perspective to alter this alienating direction. For this reason, open source's most popular contributions are in areas far divorced from users. Developers too often create products with a target audience outside a set of highly-proficient users and fail to appreciate differing needs. When it comes to something like an implementation of *dhcpd*, this is perfectly fine. However, when direct user interaction increases, the importance of accessibility becomes clear as described before.

A brief survey of help forums and feature-request tickets yields a plethora of quotes from developers that are deeply unsettling. During the final stages testing KDE 4's deployment, the Kiwi Project noticed that the basic feature of playing an audio CD had disappeared. The most immediate answer available, located in a forum, was at the time "we developers don't find this an important feature."[2] Apparently, in the course of the software being rewritten, a consensus was made by the developers of one of the most popular Linux audio players that 'everyone just rips CDs right away anyway.' In fact, between half a dozen audio players, whose differences seemed defined by which GUI toolkit was used to copy iTunes, in the end none of these players was complete or satisfying; none played audio CDs.[3]

The open-source model's mantra of free development is rightfully heralded by its proponents as the modern vision of massive collaboration, however its invidious anti-authority tendencies are rarely mentioned. When developers aren't constrained by hierarchy, the potential for confusion, dissension and secession increases exponentially. Unlike the commercial sector, developers are able to say "KDE 4.1 has taken KDE down a path I don't want to follow. So, I have a suggestion. Fork KDE,"[4] and spend time on their alternative vision. In effect, this creates duplicate work, diverging standards and a weaker community. This is also how a diversity of window managers, such as KDE 4.x, KDE 3.5, Gnome, XFCE, exist and split the install base of the operating system and decrease transparency. This partisanship filters downward when developers are forced to pick toolkits and invest themselves in the success of one side. Similar predicaments exist for word-processing (OpenOffice, KOffice, Abiword), web browsing (Firefox, Konqueror), audio and photo management.

## 3. BEING USEFUL

For all these enumerated reasons, Linux pushes the Kiwi Project into being a user-support program to rectify educational deficiencies and lack of infrastructure. As is the nature with older hardware, components are much more prone to failure, the organization has to be around to support technical issues. It cannot be solely the responsibility of project such as Kiwi to overcome the disempowerment created by propriety software and specific vocational training.

Still, were it feasible, the returns brought to the users in alternative models like F/OSS are intangible and, thus, there is a low incentive for switching. Concern over viruses and intellectual property is a luxury; a computer that is rarely turned on is fairly immune to viruses. To a middle-aged, new computer-user it doesn't matter how open an xml document-type is, if an employer can't open it in Office.

There are certainly places where F/OSS offers competitive advantages for end-users, namely Firefox and OpenOffice. For our audience, however, Linux cannot compete with Windows when it is offered at low cost, free or pirated. Any organization solely promoting Linux is then forced to resort to coercive measures such as abandoning support if users switch away. Instead similar organizations should pragmatically be willing to consider making commercial software available, when it is offered competitively and compatible with the goals of the organization.

Continuing to espouse free platforms solely on the virtue of its collaborative origins and security is weak. To be sure, many ideological battles are legitimate, some are even existential threats. It is absolutely fundamental to the longer-term competitiveness and survival of F/OSS that the community be actively engaged in fighting software patents and

[1]This has been an analogy frequently brought up in F/OSS's defense in various panels and electronic conversations.

[2]Original quote has since been removed, however citation and debate is archived at: http://bit.ly/5VG5FU

[3]While there is something to be said for copying iTunes as being transparent, these players were still terribly incomplete. This, too, is only partially the fault of developers due to restrictions of the use of the mp3 codec and connecting to hardware such as iPods. However, there is no reason for all these players to exist other than the split between QT/GTK.

[4]Standing and productivity of author mean less than the support and sincerity thrown to the notion. http://bit.ly/82VrNU

the proliferation of closed standards. However, it remains the concern of the Kiwi Project that blindly promoting the current slate of open source software, without acknowledging the aforementioned shortcomings is reckless and irresponsible.

In the long term, there are clearly differences that educators and advocates can make to the problems that the Kiwi Project has encountered. In the least there is a need to foster a sense of focus and trusteeship towards the needs of the broader computing community, and not simply the needs of self or employer. Interestly, such problems have been addressed in a previous interation of this conference.[2] What the author found was a shocking lack of social awareness in their community of computer science students. From personal experience, this detachment appears to be a common flaw. The principle of the Kiwi Project as a student-led community/software project[1] represents one model of addressing these issues. It is unlikely that any one concept will force all students out of the malaise that is frequently discussed. However, by forcing personal interaction between future members of the developer community and neglected portions of the public, students are forced to discover these disparities for themselves, see the results and, hopefully, adjust their mindset for the future.

# 4. REFERENCES


[1] Anderson, C., and Stokke, T. Impediments to ubiquitous, free computing. In *Midwest Instruction and Computing Symposium* (April 2009).

[2] Horstmann, C. S. Challenges and opportunities in an open source software development course. In *Integrating FOSS into the Undergraduate Computing Curriculum, Free and Open Source Software (FOSS) Symposium* (March 4th 2009).