Innovation in Open Source Software Development: A Tale of Two Features

John Noll

Santa Clara University
Computer Engineering Department
500 El Camino Real, Santa Clara, CA USA
jnoll@cse.scu.edu

Abstract. Open Source Software Development appears to depart radically from conventional notions of software engineering. In particular, requirements for Open Source projects seem to be asserted rather than elicited.
This paper examines two features of selected open source products: "tabbed browsing" as realized in the Firefox web browser, and "edge magnetism" found in the Gnome desktop environment's Metacity window manager. Using archives of mailing lists and issue tracking databases, these features were traced from first mention to release, in attempt to discover the process by which requirements are proposed, adopted, and implemented in their respective Open Source projects. The results confirm the importance of user participation in Open Source projects.

1 Introduction

A common view states that open source projects begin as the need for "scratching a developer's personal itch [20]," in other words, to fill a need that is not addressed by a current commercial or free product. This lead developer then becomes the shepherd of a growing community of volunteers who contribute programming labor to the project until it evolves into a useful product. But once started, how do open source projects continue to innovate? How do they stay competitive with, and even dominate, their commercial competition? Empirical studies of open source software development suggest that open source projects follow different processes than traditional textbook approaches [24].

This paper examines the history of two features of two open source software products — "tabbed browsing" in the Mozilla project's Firefox web browser, and the "edge magnetism" behavior of the Gnome desktop environment's Metacity window manager — to see how new features are proposed, debated, and finally adopted for release. This examination shows that innovation occurs in a variety of ways, sometimes following a conventional software engineering approach, other times resembling Raymond's itch scratching. Perhaps most interestingly, these differing processes can coexist in a single project, providing multiple sources for innovation.

Krishnamurthy's study of projects hosted by *Sourceforge.net* supports Raymond's idea of project initiation: of the 100 most active projects marked *mature* in the Sourceforge coding scheme, the median number of developers on a project was four [12],

confirming that many projects start small, but also suggesting that many projects never grow beyond a handful of developers.

But the itch scratching explanation does not fit all open source projects, nor does it explain how at least some open source software grows and evolves past the initial release into feature-rich products; many open source projects continue to evolve into comprehensive products that have capabilities far beyond their original conception. They have excellent quality and sometimes dominate their markets.

For example, the Apache web server has grown to include numerous innovative features including a built-in Java virtual machine, Perl interpreter, and database access, that extend Apache's functionality far beyond its original purpose as an HTTP server of HTML home pages. The Apache server is reported to host the majority of the world's web sites [29, 15]. The Apache foundation now includes projects far beyond the Apache web server product, encompassing such diverse elements as XML processing libraries and parsers, software build tools, and email processing software [5].

Open source projects based on formerly proprietary products also continue to innovate. The Firefox web browser is considered to be one to the most secure web browsers available and is actually gaining market share compared to its chief commercial competitor (Internet Explorer) [10, 22]. Firefox evolved from the Mozilla codebase, which in turn is a descendant of the Netscape Navigator code that was released as open source by Netscape Communications, Inc. [6]

Similarly, OpenOffice was created as an open source version of Sun Microsystem's StarOffice commercial office automation product [18]; it now has many innovative features including an XML-based storage format and plug-in "channels" for importing other file formats. In both cases, these products have evolved into feature-rich products that take them far beyond mere copies of their commercial competitors.

The goal of this paper is to discover how mature Open Source Software projects develop new features. Toward this end, the history of Firefox's tabbed browsing and Metacity's edge magnetism was traced by examining discussions of each feature in project mailing lists, web logs, issue tracking systems, and other on-line forums. The results provide a snapshot of how new ideas are incorporated into products, providing further insight into Open Source development processes.

The next section describes the projects and features studied. Following that is a discussion of observations and their implications. The last sections present related work and conclusions.

2 Background

Two open source software products were chosen to study: the Firefox web browser, and the Metacity window manager for the Gnome desktop environment. These projects represent different types of open source projects: Firefox has roots in a proprietary product (Netscape Navigator), while the Gnome project was open source from its inception. Firefox has at least one serious proprietary competitor, while Gnome is targetted for Unix and Linux platforms, and thus its chief competitor is another open source

product (KDE). The Firefox architecture incorporates extension mechanisms that allow programmers to add new functionality without modifying the core source code. Gnome, being an environment for managing multiple applications on a user's desktop, provides services and libraries that programmers use to create applets and applications that may cooperate with other Gnome programs. Metacity is one such program, created with simplicity as a goal and thus providing minimal customization options.

These two projects also have significantly different organization and management structure. The Mozilla development organization has a substantial co-located workforce that can hold traditional face-to-face meetings; these are used for release planning. Gnome is a "pure" open source project with a governing board and foundation to accept contributions from commercial firms. The project management and labor remain completely distributed. Open Office falls in between these extremes: it receives significant support from Sun microsystems, including funds and labor, but the project management and programming effort are widely distributed.

In order to understand how each project develops new functionality, a single feature of each product was chosen from a current release:

1. "Tabbed browsing" in the Firefox web browser, which allows users to open and manage multiple web pages within a single browser window.
2. "Window magnetism" (also called "edge resistance" and "window snapping") in Gnome's Metacity window manager, which helps users place windows on the desktop in a "tiled" arrangement.

Archives of project discussion forums were then examined to determine when the feature was first proposed, how it was debated, and when it was ultimately adopted to be delivered in a specific release. Because a typical open source project involves widely distributed programmers, testers, and users, management and technical discussions are conducted using digital communication technology such as email lists, news groups, issue tracking systems, and increasingly "chat" channels and web logs. With the exception of chat channels, these discussions are archived and made available to the general public, as a way of preserving the history of design decisions and to provide a means for newcomers to understand how a given project conducts its business. Discussions conducted via chat channels are sometimes archived as well, but this practice does not seem to be as common as archiving other media.

The details of each feature are explained in the following sections.

2.1 Firefox Tabbed Browsing

Firefox is a web browser developed by the Mozilla foundation. The Mozilla foundation was created to oversee the continued evolution of Netscape Navigator and Communicator when Netscape Communications, Inc. decided to transition development of their web browser and related software to an open source model. The Mozilla foundation's orginal product, *Mozilla*, is an integrated web browser, email client, and web page composer; while it is still being distributed and maintained, the Mozilla foundation's long term strategy is to replace the monolithic Mozilla product with separate, single-purpose programs: the Firefox web browser, and the Thunderbird email client [1].

Firefox has been highly successful, earning praise for its innovative features as well as robustness [10, 22].

Fig. 1: Tabbed Browsing in Firefox

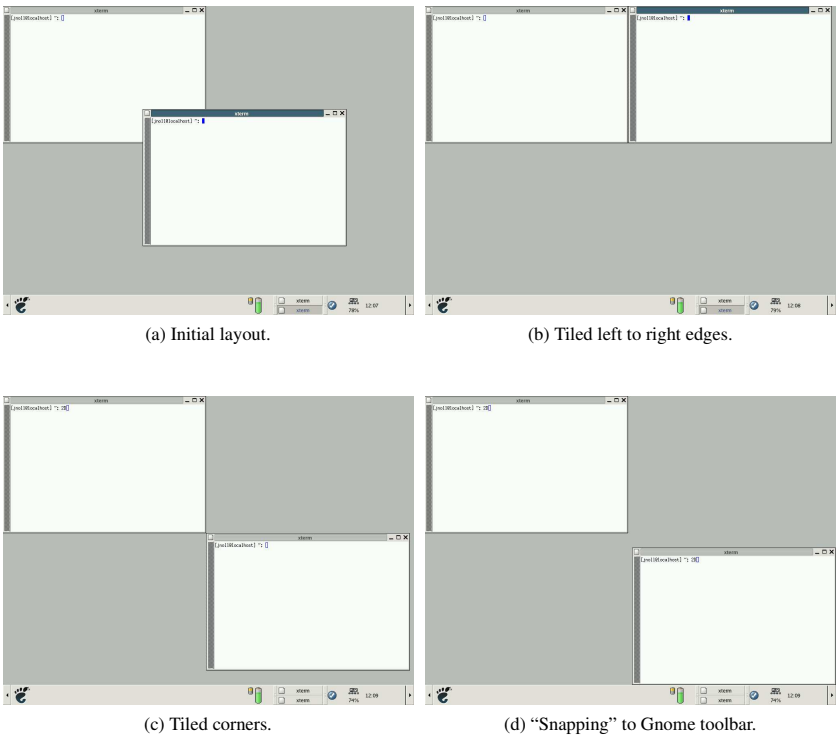
Tabbed browsing is a feature available in several comtemporary web browsers that allows the presentation of multiple web pages in a single window (Figure 1). Each page is identified by a "tab" resembling the label tab of a paper file folder; users can switch the window's display from one page to another by clicking on the page's associated tab. In the current version of Firefox (version 1.5 as of this writing), the pages can be re-ordered by dragging the tabs to the right or left; new pages can be opened in an existing tab, a new tab, or a completely new browser window.

2.2 Metacity Edge Magnetism

Metacity is the default window manager for the Gnome desktop environment. Unlike Microsoft windows and other window-based user interfaces that are part of the operating system, the X Window System — the windowing platform on which Gnome runs — is a set of user-space programs that work together to create the windowing environment. A *window manager* for the X window system is the program that is responsible for creating windows and decorating them with borders and buttons to minimize, maximize, and close windows; the window manager is also responsible for moving windows in response to mouse or keyboard events, and to provide keyboard shortcuts. As such, the window manager has a significant effect on the appearance and operation of a user's desktop.

Because a window manager is separate from the operating system, users are free to choose any window manager that suits their needs and taste; a variety of window managers for the X window system have been created to satisfy different user requirements.

Metacity was created as a replacement for Sawfish, the previous default window manager for the Gnome desktop environment. Metacity could be seen as a reaction to Sawfish's complexity and lack of stability; Sawfish was highly configurable, having a built-in Scheme interpreter, but had a high fault rate which many considered to be a side effect of its rich functionality. This excerpt from a posting to the Gnome support forum illustrates [3]: "... people praise Sawfish features yet they hate the massive amount of bugs. These two things go hand in hand. There is a reason Sawfish is practically not maintained anymore." Metacity was designed to include the minimum set of useful features, with minimal configurability; the emphasis was on robustness rather than richness.



(a) Initial layout.

(b) Tiled left to right edges.

(c) Tiled corners.

(d) "Snapping" to Gnome toolbar.

Fig. 2: Window edge magnetism in Metacity.

Window *edge magnetism*, also called *edge resistance* or *snapping*, makes positioning a window on the desktop easier by changing the behavior of a window when it is moved near another window: the moving window will resist overlap of another window, and will try to align its edges with that window (Figure 2). The windows behave as if they were magnetized; like real magnets, they resist certain arrangements and "snap" to a tiled arrangement where window sides (Figure 2b) and corners (Figure 2c)

are aligned, and windows do not overlap other windows. This makes organizing the desktop for maximum window visibility easier.

3 Observations

3.1 Tabbed Browsing

The first reference to tabbed browsing on a Mozilla-oriented newsgroup appears to be June 23, 1999, when a Mozilla user posted a note to `netscape.public.mozilla.wishlist` requesting the ability to download a new web page while viewing another page [16]:

> One thing that I would really want to see is the ability to open a link in the new window in background (i.e. the focus should remain in my current window, and new window should load silently, without bothering me until it is ready and I am ready to read it).

This spawned a discussion of the merits of tabbed browsing; the following poster is referring to the Opera web browser's tabbed browsing feature: [7]

> Have you tried tabbed browsing? Now that I've tried it, I won't go back to windows everywhere. The idea is that pages have their own tabbed windows. Instead of juggling windows, you just click their tabs. The beauty part is new pages open in the background, just as you requested. The tab tells you when the page is done loading. Then you just click over. Shweet!

H.J. van Rantwijk claims to have proposed addition of tabbed browsing to Mozilla on the Mozilla developer's chat forum (`#mozilla` at `irc.mozilla.org`), but got no positive response. The Mozilla foundation does not make public its archives of chat channels, so this claim is difficult to verify. Regardless, using Mozilla's extension mechanism, he was able to implement and distribute this functionality anyway. The result, an extension called MultiZilla, implemented the first tabbed browsing functionality for the Mozilla browser. This project began in April 2000 [28].

Subsequently, David Hyatt implemented tabbed browsing for Mozilla (version 0.97, released in December, 2001) directly, influenced by MultiZilla. But this implementation was done from scratch without using any code from MultiZilla [27]. Hyatt went on to create the Firefox browser (with Blake Ross); the first implementation of Firefox (then called "Phoenix") to feature tabbed browsing was the 0.3 milestone of October, 2002, which led to the first official Firefox product release (1.0) in November of 2004 [9].

What this suggests is that new features can follow several paths from suggestion to release. Tabbed browsing first appeared in Mozilla as an extension written by a volunteer who was unsuccessful getting acceptance from the core Mozilla developer community. The extension proved to be useful enough that one of the core developers incorporated it into the main Mozilla code base, which eventually led to its inclusion in Firefox.

An important aspect of this path is that it is enabled by the Firefox architecture: because Firefox's user interface implementation allows user interface behavior to be defined using a specification language called XUL [2], writers of extensions to the user interface don't have to change any of the core code; rather, they write a new XUL specification.

3.2 Edge Resistance

Edge resistance was available in other window managers, including the existing Gnome window manager (Sawfish), when Metacity was released. Many users missed this feature in Metacity; the following quote[1] from a posting to the `gnomesupport.org` forum illustrates [13]

> Recently, I updated my box to Redhat 9.0 and dropped Sawfish in favour of Metacity. However, there are two things I used to use in Sawfish I am not able to use in Metacity:
> - Configure keys to move the cursor.
> - Switch on windows "magnetism" to help a easy windows placement.
> I didn't find any option, does anybody here know where to touch?

This comment sparked a debate on the GNOME support forum (gnomesupport.org) centered on the tension between feature richness and maintainability. Another poster echoed the above sentiments [30]:

> Fly has a point about the usability of Metacity. I understand the complaints about the "bloat" in Sawfish, but as far as memory footprint is concerned, there is very little difference between Sawfish and Metacity. To claim that including sensible features is adding bloat is just feeding us a line of bullshit.
> I've been using Metacity since Gnome 2.0, mainly because it is now inconvenient to manage themes properly with Sawfish in the picture, but it would be very nice if Metacity would remember window sizes and placement. As far as I'm concerned, that is a window manager's job, and Metacity is clearly shirking that job.
> If it wasn't for those apps that remember their own window geometry, I would be getting quite fed up with Metacity by now.

Then, "Fly" followed his earlier posting with some general comments

> ... I understand that many features in Sawfish [are] excessive or unrelated to WM, but 80% of Sawfish features very useful and I need it - you not?

In response, "Dbrody" (a 'guru' on this forum) said

> But if only $> 5\%$ of people need $> 80\%$ of those features then you have just proved that it is bloat. Bloat dosn't mean memory foot print. That is NOT what anybody cares about the extra 1k of ram. Bloat means the maintainer needs to start maintaining more features. More bug reports. More tweaking of those

[1] Original spelling and proper name capitalization in quoted excerpts has been corrected.

> advanced features. etc... etc... Metacity is not even 1.0 yet. There are many changes that are planned to go into Metacity but haven't yet because things go a little slowly, or because it will make Metacity incompatible with themes and so forth.
> Also, many of these feature can be done with external programs, like devilspy. Certainly things like edge flipping, advanced focus management, etc... are easily done using libwnck and a little hacking.

This debate is interesting because it takes place in a public forum where anyone - users, developers, interested third parties - can participate. The discussion of requirements is therefore completely transparent, and also recorded in significant detail, so that the rationale behind any decision can be discovered later if necessary.

The creator of Metacity agreed on the usefulness of edge magnetism almost a year earlier; he filed the following Request for Enhancement (RFE) in the Gnome project's issue tracking system in May 2002 [19]:

> Add some kind of mild "attraction" to window/screen edges, perhaps only after a timeout. Need to experiment.

This entry stimulated a lengthy discussion of exactly how this behavior should work, which continued for over three years until the feature was incorporated into the release code-base in November of 2005.

Again, the discussion takes place in a public forum (the Gnome issue tracking database is readable by anyone, and anyone who registers can post issue reports or comments), and is recorded for future reference.

The history of edge magnetism in Metacity represents a combination of two phenomena that appear to be characteristics of Open Source development projects. First, the significant, lively participation by users of Metacity in the discussion about the merits of and desire for edge magnetism are an example of the essential role that users play in Open Source projects [4]. Second, Havoc Pennington's RFE is an example of an asserted requirement: the developer of a product has stated the need for a particular feature; this is consistent with observations made by other researchers [8, 23].

3.3 Discussion

Gnome and Metacity closely resemble the common notion of open source development, where features are proposed in an on-line forum (newsgroup, mailing list, issue database), debated by users and programmers, and ultimately adopted or rejected. A feature may be adopted by virtue of having a working implementation, regardless of its merits.

In contrast, Firefox follows an almost traditional process involving regular face-to-face release planning meetings. But Firefox's extension mechanism allows features that are initially rejected to "prove" their worth by demonstrating adoption by real users.

The openness of various communication channels employed by open source projects enables and encourages enthusiastic participation by users of the product, as well as

developers. This provides early feedback about a product's functionality and shortcomings, as well as a way to capture users' ideas and needs.

Likewise, open issue tracking mechanisms provide a way for end users to voice their concerns about product functionality, and participate in the discussion about resolutions and enhancements. This has advantages for both the developers and users: the developers can potentially seek clarification through the discussion feature of issue tracking systems like Bugzilla, while users seem to develop a sense of ownership as they see their concerns actively considered and their participation encouraged.

4 Related Work

Studies of open source software projects address a wide range of topics from economics [29] to maintainability [25].

A number of case studies have examined open source development processes, including those employed by Apache and Mozilla [14, 21, 24]. In particular, Reis and de Mattos Fortes, in their study of Mozilla development processes, report that high level requirements are specified by the `mozilla.org` management, but all development on the Mozilla code base originates with a "bug" report, which might be submitted by another developer, tester, or end user [21]. These reports may document some product failure, or a request for enhancement.

Feller and Fitzgerald note that users are a "critical feature" [4, 10] of OSSD projects, as the source of new requirements. Scacchi has made several studies of requirements acquisition in open source software development; he observes that requirements "emerge" from on-line discussions which are usually open forums, rather than through traditional requirements elicitation processes, but that this emergent process, though less formal, is also effective [24, 23]. He also notes that requirements are "asserted" after the fact; other researchers have echoed this observation. In particular, German reports a similar situation in the Gnome project [8]. This seems to contradict conventional understanding that cites failure to understand requirements as a major source of software project failure.

But Trudelle notes, in his discussion of lessons learned from experience working on Mozilla, that this approach led to rework of some of the Mozilla implementation in response to user-submitted bug reports; his view is that this rework could have been avoided with traditional up-front requirements analysis and design activities [26]. Henderson echoes this view, claiming that open source projects do not employ "requirements elicitation," but that this could (and should) be easily added to open source processes [11]. Further, Nichols and Twidale observe that usability requirements are not captured well by OSSD projects, due to the mismatch between developers and users; their view is that the OSSD approach of "coding as early as possible" violates "good interface design." [17]

These observations run counter to the prevailing OSSD view that de-emphasizes formal design and requirements gathering. Trudelle's view — that OSSD projects need an overarching UI design and design function — seems to contradict the current success of Firefox, which is widely recognized as among the most innovative web

browsers. In particular, Nichols and Twidale's assertion that "commercial software establishes the state of the art" [17] seems to be contradicted by Opera and Firefox, both of which included UI features (tabbed browsing, for example) well before Internet Explorer.

5 Conclusions

Much has been made of the advantages open source development might give to commercial for-profit enterprises, including high quality, free labor, and quick response to critical failures. But the observations presented above reveal some practices that could be useful to any software development effort, including traditional closed source products:

1. Open communication channels between users and developers. This gives users a greater stake in the future of the product, and provides feedback without the overhead of conducting surveys or convening focus groups.
2. Extension mechanisms that allow users with programming skills to demonstrate ideas by contributing working functionality.
3. Alternate paths for ideas to become released features.

Open source projects are far from uniform in their process for conceiving and realizing new features. But they seem to share a common aspect — close involvement of end users in the development process — that is less common in conventional development environments.

Acknowledgments

This work was supported in part by a grant from the School of Engineering at Santa Clara University. No endorsement is implied.

References

1. Alex Bishop. Major roadmap update centers around Phoenix, Thunderbird; 1.4 branch to replace 1.0; changes planned for module ownership model. *MozillaZine (online)*, April 2 2003. `http://www.mozillazine.org/articles/article3042.html`.
2. Peter Bojanic. The joy of XUL. Web page, cited september 6, 2006., Mozilla Foundation, June 2006. `http://developer.mozilla.org/en/docs/The_Joy_of_XUL`.
3. Dbrody. no title. `http://gnomesupport.org/forums/viewtopic.php?t=3603&highlight=&sid=c5f4%e5ae34765db22bac227d7f8b17cb`, September 2003. Posting to the Gnome desktop user support forum.
4. Joseph Feller and Brian Fitzgerald. A framework analysis of the open source software development paradigm. pages 58–69, 2000.
5. The Apache Software Foundation. About the Apache HTTP server project. `http://httpd.apache.org/ABOUT_APACHE.html`. Web page, cited January 16, 2007.

6. The Mozilla Foundation. About the Mozilla foundation. http://www.mozilla.org/foundation/, November 2006. Web page cited January 16, 2007.
7. Gboone. Open new window in background (tabbed browsing). http://groups.google.com/group/netscape.public.mozilla.wishlist/tree/br%owse_frm/thread/ef62c3307e2a7a32/4ec071eae14082ff?rnum=1&hl=en&_done=%2Fgroup%%2Fnetscape.public.mozilla.wishlist%2Fbrowse_frm%2Fthread%2Fef62c3307e2a7a32%2F%4ec071eae14082ff%3Ftvc%3D1%26hl%3Den%26#doc_4b33ef52c30564cf.
8. Daniel M. German. GNOME, a case of open source global software development. In *Proceedings of the 6th International Workshop on Global Software Development*, Portland, OR USA, May 2003.
9. Ben Goodger. Firefox 1.0 roadmap. http://www.mozilla.org/projects/firefox/roadmap-1.0.html, 2004. Web page describing release history of Firefox, cited March 1, 2007.
10. Steve Hamm. A Firefox in IE's henhouse. *Business Week*, September 17 2004.
11. Lisa G. R. Henderson. Requirements elicitation in open-source programs. *CrossTalk - The Journal of Defense Software Engineering*, 13(7):28–30, July 2000. http://www.stsc.hill.af.mil/crosstalk/2000/07/henderson.html.
12. Sandeep Krishnamurthy. Cave or community?: An empirical examination of 100 mature open source projects. *First Monday*, 7(6), June 2002.
13. Lou. Metacity configuration. http://gnomesupport.org/forums/viewtopic.php?t=3603&highlight=&sid=c5f4%e5ae34765db22bac227d7f8b17cb, August 4 2003. Posting to the Gnome desktop user support forum.
14. Audris Mockus, Roy T. Fielding, and James Herbsleb. A case study of open source software development: The Apache server. In *Proceedings of the 22nd International Conference on Software Engineering*, pages 263–272, Limerick, Ireland, May 2000.
15. Netcraft, Ltd. September 2006 web server survey. http://news.netcraft.com/archives/2006/09/05/september_2006_web_server_%survey.html, September 2006.
16. Vladimir Neyman. Open new window in background. http://groups.google.com/group/netscape.public.mozilla.wishlist/tree/br%owse_frm/thread/ef62c3307e2a7a32/4ec071eae14082ff?rnum=1&hl=en&_done=%2Fgroup%%2Fnetscape.public.mozilla.wishlist%2Fbrowse_frm%2Fthread%2Fef62c3307e2a7a32%2F%4ec071eae14082ff%3Ftvc%3D1%26hl%3Den%26#doc_4ec071eae14082ff, June 23 1999. Message posted to netscape.public.mozilla.wishlist mailing list.
17. David M. Nichols and Michael B. Twidale. The usability of open source software. *First Monday*, 8(1), January 2003.
18. OpenOffice.org. About us: OpenOffice.org. http://about.openoffice.org/index.html, January 2007. Web page, cited January 19, 2007.
19. Havoc Pennington. Bug 81704 - Edge magnetism/resistance/snapping/etc. http://bugzilla.gnome.org/show_bug.cgi?id=81704, May 2002. Request for enhancement (RFE) entered into the Gnome project's issue tracking system.
20. Eric S. Raymond. The cathedral and the bazaar. In *The Cathedral and the Bazaar*. O'Reilly and Associates, October 1999.
21. Christian Robottom Reis and Renata Pontin de Mattos Fortes. An overview of the software engineering process in the Mozilla project. In *Proceedings of the Open Source Software Development Workshop*, Newcastle upon Tyne, UK, February 2002.
22. Rachel Rosmarin. Mozilla Firefox gaining ground on Microsoft IE. *Forbes.com*, August 1 2006.

23. Walt Scacchi. Understanding the requirements for developing open source software systems. *IEE Proceedings – Software*, 149(1):24–39, February 2002.
24. Walt Scacchi. Free and open source development practices in the game community. *IEEE Software*, pages 59–66, January 2004.
25. Stephen R. Schach, Bo Jin, David R. Wright, Gillian Z. Heller, and A. Jefferson Offut. Maintainability of the Linux kernel. *IEE Proceedings – Software*, 149(1), February 2002.
26. Peter Trudelle. Shall we dance? Ten lessons learned from Netscape's flirtation with open source UI development. Technical report, Mozilla.org, 2002. Presented at the Open Source Meets Usability Workshop, Conference on Human Factors in Computer Systems (CHI 2002), Minneapolis, MN. Accessed December 28, 2006.
27. unknown. A guide to Mozilla 1.0. `http://www.mozilla.org/start/1.0/guide/`, 2002. Web page describing release 1.0 of Mozilla.
28. H.J. van Rantwijk. MultiZilla's home page. `http://multizilla.mozdev.org`, February 24 2006. Home page for the MultiZilla project, cited September 6, 2006.
29. David A. Wheeler. Why open source software / free software (OSS/FS, FLOSS, or FOSS)? Look at the numbers! Technical report, dwheeler.com, 2005.
30. WonkoTheSane. Untitled. `http://gnomesupport.org/forums/viewtopic.php?t=3603&highlight=&sid=c5f4%e5ae34765db22bac227d7f8b17cb`, September 22 2003. Posting to the Gnome desktop user support forum.