An Introduction to Open Source Communities

Eugene Eric Kim <eekim@blueoxen.org>
Blue Oxen Associates
BOA-00007
April 2003

This report was sponsored by the Omidyar Foundation.

Acknowledgements

Several people supported this research in a variety of ways. Thanks go to TouchGraph's Alex Shapiro and SquirrelMail's Jonathan Angliss, Rick Castello, Marc Groot Koerkamp, and Jason Munro for their thoughtful comments about their respective open source projects. Thanks go to OSDN's Jeff Bates, Patrick McGovern, and Nathan Oostendorp, and Simtel.net's David Kirsch and Don Watkins for their valuable assistance in providing relevant statistics from their Web sites. Thanks go to Chris Dent, Richard Gabriel, and H. Jessica Kim for reviewing drafts of this report. Finally, special thanks go to Matt Hamilton of the Omidyar Foundation for making this report possible.

Distribution License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

1. Definitions

a. **"Collective Work"** means a work, such as a periodical issue, anthology or encyclopedia, in which the Work in its entirety in unmodified form, along with a number of other contributions, constituting separate and independent works in themselves, are assembled into a collective whole. A work that constitutes a Collective Work will not be considered a Derivative Work (as defined below) for the purposes of this License.

b. **"Derivative Work"** means a work based upon the Work or upon the Work and other pre-existing works, such as a translation, musical arrangement, dramatization, fictionalization, motion picture version, sound recording, art reproduction, abridgment, condensation, or any other form in which the Work may be recast, transformed, or adapted, except that a work that constitutes a Collective Work will not be considered a Derivative Work for the purpose of this License.

c. **"Licensor"** means the individual or entity that offers the Work under the terms of this License.

d. **"Original Author"** means the individual or entity who created the Work.

e. **"Work"** means the copyrightable work of authorship offered under the terms of this License.

f. **"You"** means an individual or entity exercising rights under this License who has not previously violated the terms of this License with respect to the Work, or who has received express permission from the Licensor to exercise rights under this License despite a previous violation.

2. Fair Use Rights. Nothing in this license is intended to reduce, limit, or restrict any rights arising from fair use, first sale or other limitations on the exclusive rights of the copyright owner under copyright law or other applicable laws.

3. License Grant. Subject to the terms and conditions of this License, Licensor hereby grants You a worldwide, royalty-free, non-exclusive, perpetual (for the duration of the applicable copyright) license to exercise the rights in the Work as stated below:

a. to reproduce the Work, to incorporate the Work into one or more Collective Works, and to reproduce the Work as incorporated in the Collective Works;

b. to distribute copies or phonorecords of, display publicly, perform publicly, and perform publicly by means of a digital audio transmission the Work including as incorporated in Collective Works;

The above rights may be exercised in all media and formats whether now known or hereafter devised. The above rights include the right to make such modifications as are technically necessary to exercise the rights in other media and formats. All rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly made subject to and limited by the following restrictions:

a. You may distribute, publicly display, publicly perform, or publicly digitally perform the Work only under the terms of this License, and You must include a copy of, or the Uniform Resource Identifier for, this License with every copy or phonorecord of the Work You distribute, publicly display, publicly perform, or publicly digitally perform. You may not offer or impose any terms on the Work that alter or restrict the terms of this License or the recipients' exercise of the rights granted hereunder. You may not sublicense the Work. You must keep intact all notices that refer to this License and to the disclaimer of warranties. You may not distribute, publicly display, publicly perform, or publicly digitally perform the Work with any technological measures that control access or use of the Work in a manner inconsistent with the terms of this License Agreement. The above applies to the Work as incorporated in a Collective Work, but this does not require the Collective Work apart from the Work itself to be made subject to the terms of this License. If You create a Collective Work, upon notice from any Licensor You must, to the extent practicable, remove from the Collective Work any reference to such Licensor or the Original Author, as requested.

b. You may not exercise any of the rights granted to You in Section 3 above in any manner that is primarily intended for or directed toward commercial advantage or private monetary compensation. The exchange of the Work for other copyrighted works by means of digital file-

sharing or otherwise shall not be considered to be intended for or directed toward commercial advantage or private monetary compensation, provided there is no payment of any monetary compensation in connection with the exchange of copyrighted works.

c. If you distribute, publicly display, publicly perform, or publicly digitally perform the Work or any Collective Works, You must keep intact all copyright notices for the Work and give the Original Author credit reasonable to the medium or means You are utilizing by conveying the name (or pseudonym if applicable) of the Original Author if supplied; the title of the Work if supplied. Such credit may be implemented in any reasonable manner; provided, however, that in the case of a Collective Work, at a minimum such credit will appear where any other comparable authorship credit appears and in a manner at least as prominent as such other comparable authorship credit.

5. Representations, Warranties and Disclaimer

a. By offering the Work for public release under this License, Licensor represents and warrants that, to the best of Licensor's knowledge after reasonable inquiry:

i. Licensor has secured all rights in the Work necessary to grant the license rights hereunder and to permit the lawful exercise of the rights granted hereunder without You having any obligation to pay any royalties, compulsory license fees, residuals or any other payments;

ii. The Work does not infringe the copyright, trademark, publicity rights, common law rights or any other right of any third party or constitute defamation, invasion of privacy or other tortious injury to any third party.

b. EXCEPT AS EXPRESSLY STATED IN THIS LICENSE OR OTHERWISE AGREED IN WRITING OR REQUIRED BY APPLICABLE LAW, THE WORK IS LICENSED ON AN "AS IS" BASIS, WITHOUT WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED INCLUDING, WITHOUT LIMITATION, ANY WARRANTIES REGARDING THE CONTENTS OR ACCURACY OF THE WORK.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY APPLICABLE LAW, AND EXCEPT FOR DAMAGES ARISING FROM LIABILITY TO A THIRD PARTY RESULTING FROM BREACH OF THE WARRANTIES IN SECTION 5, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

7. Termination

a. This License and the rights granted hereunder will terminate automatically upon any breach by You of the terms of this License. Individuals or entities who have received Collective Works from You under this License, however, will not have their licenses terminated provided such individuals or entities remain in full compliance with those licenses. Sections 1, 2, 5, 6, 7, and 8 will survive any termination of this License.

b. Subject to the above terms and conditions, the license granted here is perpetual (for the duration of the applicable copyright in the Work). Notwithstanding the above, Licensor reserves the right to release the Work under different license terms or to stop distributing the Work at any time; provided, however that any such election will not serve to withdraw this License (or any other license that has been, or is required to be, granted under the terms of this License), and this License will continue in full force and effect unless terminated as stated above.

8. Miscellaneous

a. Each time You distribute or publicly digitally perform the Work or a Collective Work, the Licensor offers to the recipient a license to the Work on the same terms and conditions as the license granted to You under this License.

b. If any provision of this License is invalid or unenforceable under applicable law, it shall not affect the validity or enforceability of the remainder of the terms of this License, and without further action by the parties to this agreement, such provision shall be reformed to the minimum extent necessary to make such provision valid and enforceable.

c. No term or provision of this License shall be deemed waived and no breach consented to unless such waiver or consent shall be in writing and signed by the party to be charged with such waiver or consent.

d. This License constitutes the entire agreement between the parties with respect to the Work licensed here. There are no understandings, agreements or representations with respect to the Work not specified here. Licensor shall not be bound by any additional provisions that may appear in any communication from You. This License may not be modified without the mutual written agreement of the Licensor and You.

1. Introduction

The premise of this report is that open source software communities are one of the most successful—and least understood—examples of high-performance collaboration and community-building on the Internet today. Other types of communities could benefit enormously from understanding how open source communities work.

This report describes what open source communities are and how they work. In particular, it addresses the following questions:

- What is the open source landscape as a whole? How many projects exist, what kinds of software do these projects develop, and how many people are involved with these projects?
- What are the demographics of those who participate in these communities? Why do they join, and how long do they stay? How do they interact with each other?
- How do open source communities work? What are the patterns of collaboration within successful open source communities?

In examining these questions, this report discusses existing, relevant research, and presents original case studies of two open source projects: TouchGraph (http://www.touchgraph.com/) and SquirrelMail (http://www.squirrelmail.org/). It then identifies some patterns of collaboration that both of these projects share, and describes how these patterns might apply to other types of communities. Finally, it reviews what is not yet well understood about open source communities, and proposes several paths for further research.

1.1. Open Source Communities

Source code defines the functionality of a software application. It consists of a series of instructions, written by programmers in a programming language, that gets translated by a special program (called a "compiler") into a runnable software application. Source code is akin to the blueprint of a building or the recipe of a favorite dish, with one major difference: Converting source code to software is easy and automatic. In essence, source code is the software.

"Open source" describes software whose source code may be freely modified and redistributed with few restrictions.[1] Strictly speaking, the terms of the distribution license are the only factor that determines whether or not software is open source. However, open source has come to represent much more. Over the past five years, people have become fascinated by the tremendous impact that various open source software has made in the marketplace. Most notably, the open source Linux operating system and Apache Web server have successfully challenged Microsoft's market dominance to the point where Microsoft considers these tools to be among its most serious competitors (Judge).

More importantly, open source software tends to be developed by loosely organized, ad-hoc communities consisting of contributors from all over the world who have never met face-to-face and yet who share a strong sense of commitment. Somehow, this mish-mash of people coheres to effectively accomplish an extremely complex task: building high-quality software. The success of open source software has forced people to reconsider their traditional views on software development, individual psychology, and organizational dynamics.

[1] See Open Source Initiative's "The Open Source Definition" for a more detailed definition of "open source."

2. The Open Source Landscape

2.1. The Market

There are three major databases of open source software available on the Internet today: the GNU Free Software Directory (http://www.gnu.org/directory/), SourceForge (http://sourceforge.net/), and Freshmeat (http://freshmeat.net/). The first is maintained by the Free Software Foundation (FSF, http://www.fsf.org/), and lists only projects that are available under the General Public License (GPL, http://www.fsf.org/licenses/licenses.html#GPL). The Open Source Development Network (OSDN, http://www.osdn.com/) owns the latter two repositories. Examining the projects listed in these databases offers some idea of the number and types of open source software available today. Table 2.1 lists the statistics for these three sites.

Repository	Number of Projects
SourceForge	58,245
Freshmeat	27,514
GNU Free Software Repository	2,077

Table 2.1. Number of projects listed at Freshmeat, SourceForge, and the GNU Free Software Directory. (March 2003)

Of the three sites, Freshmeat covers the broadest spectrum of open source software, and is likely to include packages listed on both SourceForge and the GNU Free Software Directory. Not all of Freshmeat's listings are open source (about 2,000 of the 27,514 projects), and a very small number of its listings describe documentation projects, not software. Many open source projects are not listed on any of these sites, although Freshmeat tracks the majority of high-profile projects.

Of Freshmeat's 20 most popular projects, 15 are intended for end users, and the remaining five are targeted towards developers and system administrators. However, of those 15, 10 run only on UNIX platforms, and the remaining five run on both UNIX and Windows. None of the 15 is written exclusively for Windows.

There are many rich and useful Windows applications that are available for free, but they continue to be predominantly shareware or freeware. Simtel.net (http://www.simtel.net/) has long been one of the largest distributors of free software for MS-DOS and Windows. However, they—along with similar sites—use the Portable Application Description (PAD, http://www.asp-shareware.org/pad/) standard for categorizing its applications, and those responsible for the standard have not deemed open source important enough to create a category for it. Open source applications on Simtel.net are generally (and incorrectly) categorized as freeware (Watkins). Table 2.2 shows a breakdown of Windows and handheld applications on Simtel.net.

Software Category	Number of Applications
Shareware	9,720
Freeware (includes GPL)	4,372
Demo	827
Public Domain	23

Table 2.2. Number of Windows and handheld applications on Simtel.net, sorted by distribution licenses.

The reason for the heavy UNIX bias among open source projects is largely historical. Distributing source code was important for the UNIX platform, because UNIX ran on a variety of machines. If you distributed a binary version of software, it would only run on one type of machines. However, if you distributed the source code to your software, users could conceivably compile it on any machine that ran UNIX.

The same was not true for PC software. Although there were many different kinds of personal computers from 1975 to 1981, the market started consolidating with the introduction of the IBM PC in 1981. Because the IBM PC and its clones running MS-DOS began to dominate the market, there was less incentive to distribute source code. First, MS-DOS did not come with a compiler, unlike most UNIX systems, so there was actually a disincentive to distribute source code. Second, because the PC market was largely homogenous, it was feasible to distribute binary versions of software. Because of these circumstances, a different kind of free software emerged for the PC market: freeware and shareware, neither of which included any license provisions for the distribution of source code.

These historical roots are largely responsible for the open source landscape today. Most of the high-profile open source projects are infrastructural applications, things like operating systems (Linux, FreeBSD), server applications (Apache, Sendmail, BIND), and software development tools (GCC, Perl, Python). Although there are a growing number of open source applications targeting end-users (GNOME, Mozilla), the majority of these applications are UNIX-oriented, although some of these applications run on Windows as well. Additionally, in the last ten years, Web-based applications (such as SquirrelMail, described below) have evolved as a new category of application.

2.2. Demographics

In the past three years, a number of surveys have attempted to shed light on the questions, "Who develops open source software, and why?" Three in particular stand out:

- *Who Is Doing It (WIDI) Survey* (2001)
- *The Boston Consulting Group/OSDN (BCG/OSDN) Hacker Survey* (2002)
- *Free/Libre and Open Source Software: Survey and Study (FLOSS)* (2002)

The results of all three surveys are reasonably consistent, and paint a good picture of the typical open source developer:

- **Overwhelmingly male.** All three surveys reported that over 98 percent of their respondents were men (FLOSS 8, BCG 21, WIDI Part 1).
- **Predominantly Generation X.** Over 70 percent of respondents from all three surveys were between the ages of 22 and 37, with the mean age ranging from 27 to 30 (FLOSS 9, BCG 21, WIDI Part 1).
- **Concentrated in the United States and Europe.** Over 80 percent of respondents from the surveys were from either the United States or Europe. The WIDI and FLOSS surveys also asked about the residences of open source developers, and found that the majority of them are also currently living in the U.S. and Europe. The FLOSS survey differed from both the WIDI and BCG/OSDN survey in that a much larger percentage of respondents were from and lived in Europe than were from or lived in the United States (FLOSS 16-17, BCG 22, WIDI Part 1).
- **IT professionals.** Over 50 percent of those surveyed work in IT. Students ranked second in all three surveys, ranging from 20 to 30 percent of respondents (FLOSS 13, BCG 25, WIDI Part 3).
- **Mostly college and high school graduates.** Both FLOSS and WIDI report that between 33 and 46 percent of respondents have college degrees, whereas between 17 and 24 percent have only a high school degree. This is consistent with the age demographics. FLOSS reports 28 percent of those surveyed have Masters degrees, whereas WIDI reports 12 percent (FLOSS 12, WIDI Part 3).
- **Part-time participation.** Between 34 and 48 percent of those surveyed spend less than five hours a week on open source software, with a clear downward trend among respondents as the numbers of hours increase. Between 9 and 15 percent spend 20 to 40 hours a week, and 5 to 7 percent spend more than 40 hours a week (FLOSS 21, BCG 23, WIDI Part 4).

2.3. Motivations

The FLOSS report found that 46 percent of its respondents do not earn any money, either directly or indirectly, for their work on open source. 16 percent are paid to develop open source software, whereas 18 percent are paid to administer it, and 12 percent are paid to support it. 26 percent claim to have received indirect financial compensation for their work on open source, and 18 percent claim their work helped them get a job (65).

Both the FLOSS and BCG/OSDN surveys explored the motivations for working on open source software. Both found that the overriding reason that people joined and continued working on open source projects was to expand and share their knowledge. 93 percent of respondents to the BCG/OSDN survey said that "increasing their personal knowledge base" was a benefit of participation, and 48 percent said that it was the most important benefit. 79 percent of the FLOSS respondents said that they joined to "learn and develop new skills," and 50 percent said that they joined to "share their knowledge and skills" (FLOSS 45, BCG 17). Figures 2.1 and 2.2 show the overall results from both the FLOSS and BCG/OSDN reports.

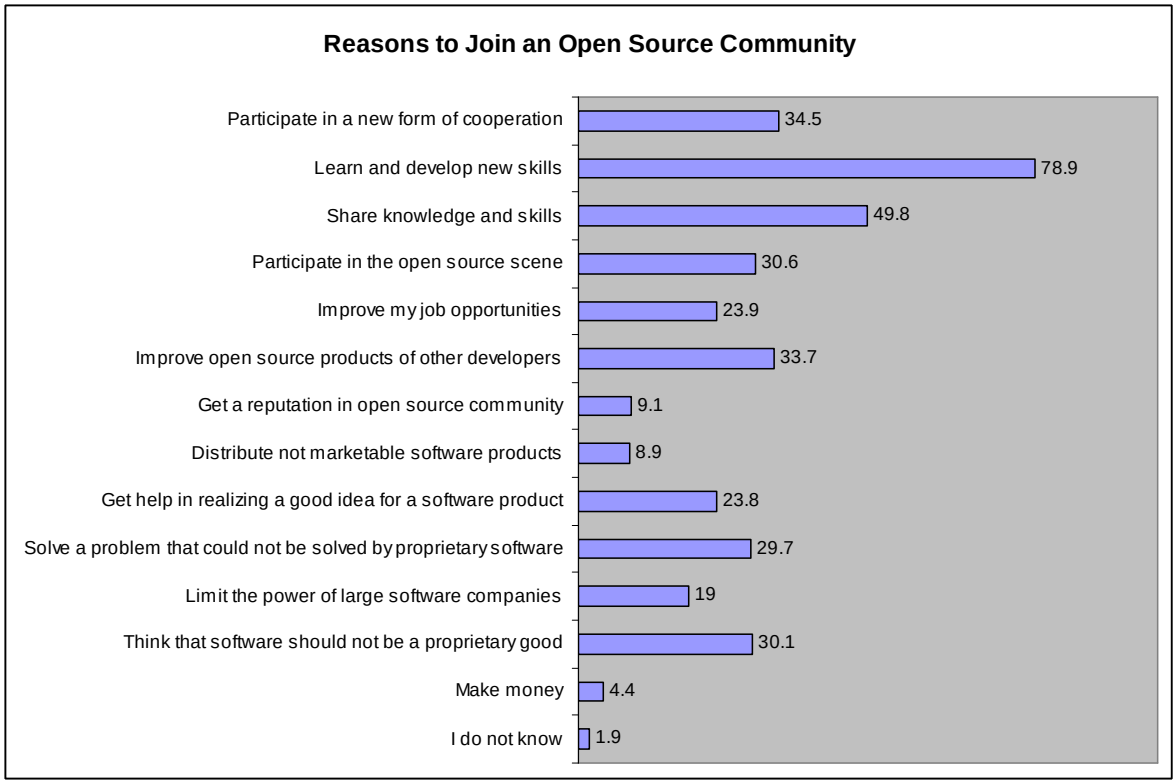

Figure 2.1. Reasons to join an open source community.

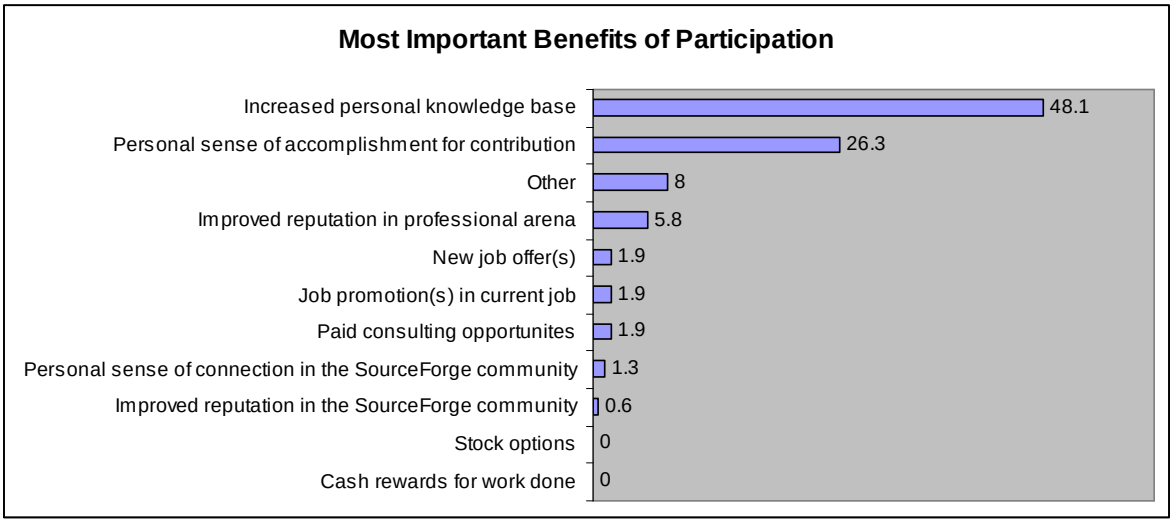

Figure 2.2. Most important benefits of participating in open source communities.

The BCG/OSDN report compared the different motivations cited by paid versus volunteer contributors. As is evident from Figure 2.3, the motivations are largely comparable, except for one—"work functionality." Those who are paid to contribute to open source software are motivated by the desire to do their job effectively (14).

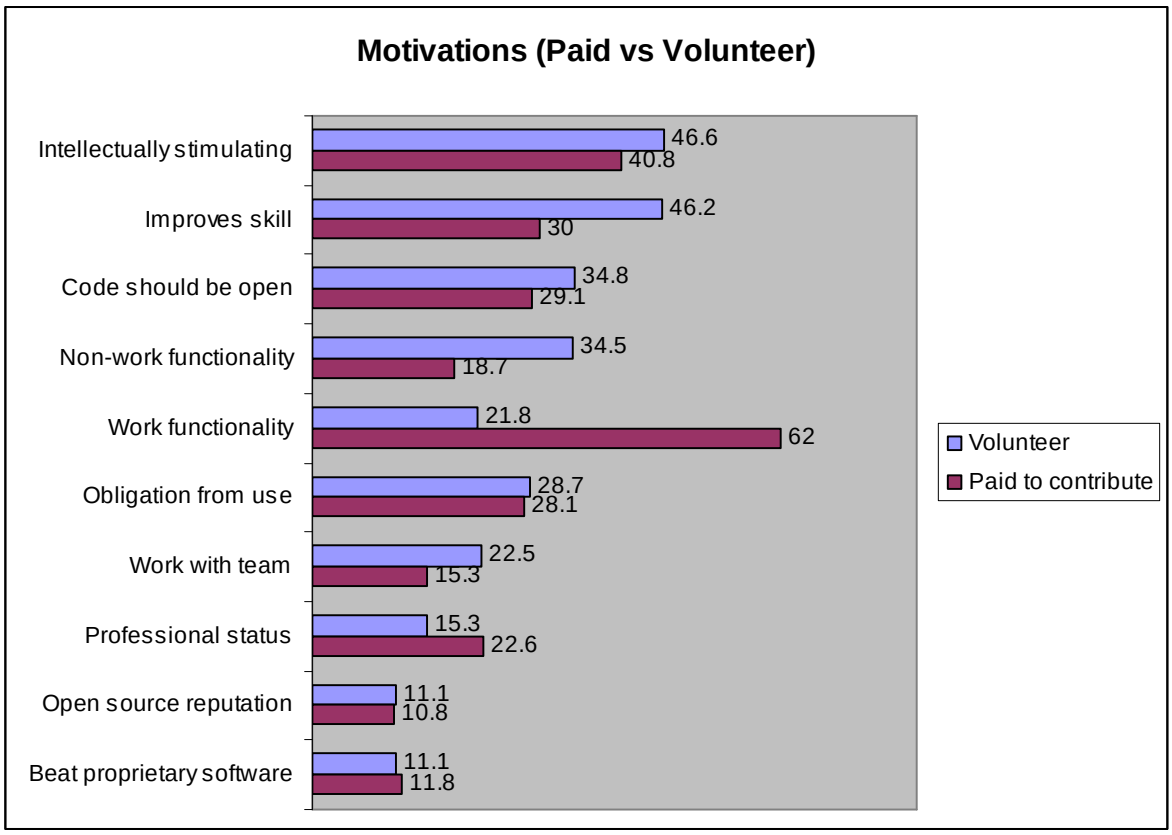

Figure 2.3. Differences in motivations between volunteers and people who are paid to do open source.

2.4. Communities

The majority of existing research on open source communities has focused on understanding the individual participants and on documenting the software development methodology. Two studies that fall in the latter category are worth noting: Sandeep Krishnamurthy's "Cave or Community? An Empirical Examination of 100 Mature Open Source Projects" (May 2002) and Audris Mockus, Roy Fielding, and James Herbsleb's *Two Case Studies of Open Source Software Development: Apache and Mozilla* (March 2002).

Krishnamurthy's report notes that the majority of open source software is actually developed by individuals, not communities. He describes this as the "lone developer (or cave) model of production." He reviewed SourceForge's top 100 most active projects, and found that the median number of developers per project was four, and the mode was one.

Mockus, Fielding, and Herbsleb did a detailed study of the development process for the Apache Web server and the Mozilla Web browser. They found that in both cases, there were core teams of developers that controlled the majority of the source code. These teams ranged in size from 10 to 15 people. They also suggested that the number of people who fix bugs is an order of magnitude larger than the size of the core team, and the number of people who report bugs are an order of magnitude larger than those who fix them (23-24).

3. Case Studies: TouchGraph and SquirrelMail

The information cited above describes the open source landscape and the individuals who develop open source software. The purpose of this section is to describe patterns of collaboration within open source communities. What are the organizational structures, processes, and tools that open source projects use to develop software, and how do these emerge? Why do people join these projects, and how long do they stay? How do the different members of these communities interact with each other, and why do they choose these methods? I explore these questions by examining two open source projects: TouchGraph and SquirrelMail.[1]

TouchGraph and SquirrelMail are both successful open source projects in their own rights, but they also represent two very different communities in many ways. The following case studies explain these differences and analyze patterns shared by both projects.

3.1. TouchGraph

TouchGraph is a software component for visualizing networks of information. Written in Java, TouchGraph allows users to navigate around an interactive graph of nodes, representing information nuggets and interconnecting relationships.

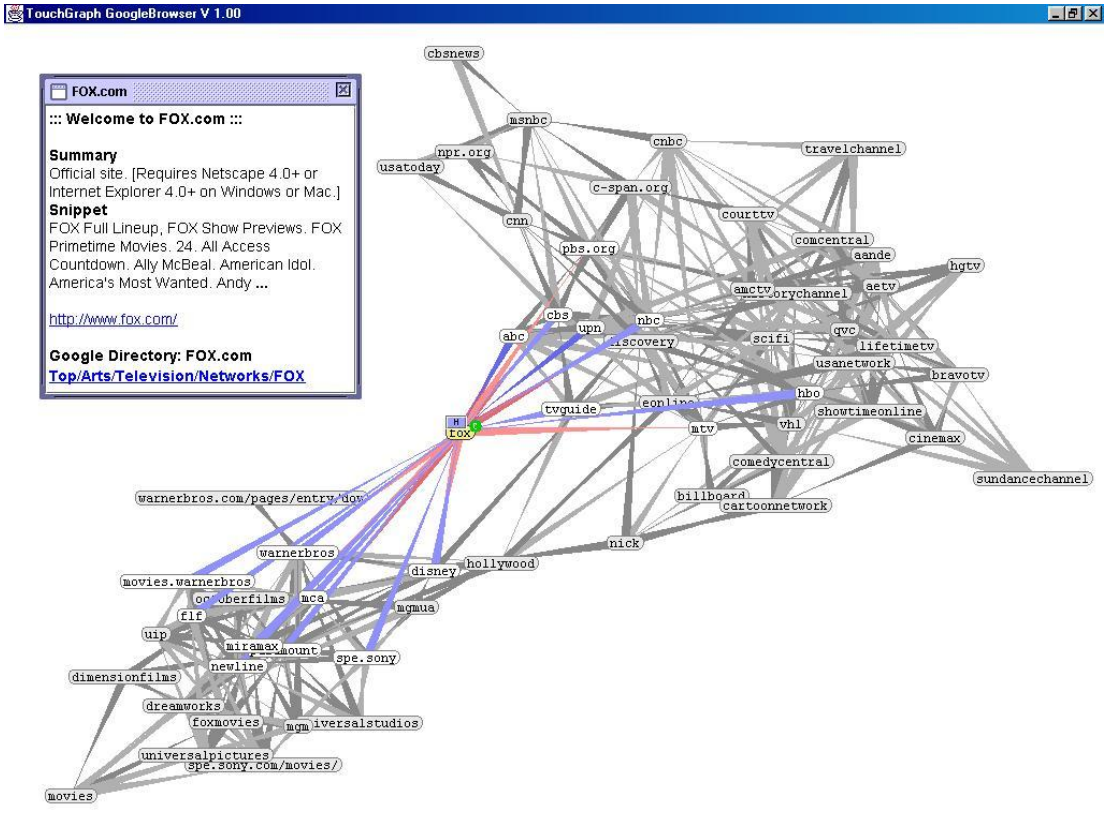

Figure 3.1. The TouchGraph GoogleBrowser (http://www.touchgraph.com/TGGoogleBrowser.html).

[1] These case studies are based on analysis of both projects' online discussion archives and on interviews with several of the projects' active participants. The project descriptions were reviewed by the projects' leaders for accuracy.

TouchGraph is not a stand-alone application. It is meant to be embedded in other applications by software developers. Perhaps the best known TouchGraph application is the TouchGraph GoogleBrowser, which graphically displays the results of a Google search, and which has been featured in numerous publications, both print and online.[1] TouchGraph is also used to visualize information in collaborative applications, from blogs to Wikis.

Alex Shapiro, TouchGraph's creator, started developing TouchGraph while working at Sapient in 2000, after graduating from Columbia Engineering School with a degree in computer science. Shapiro was interested in language and artificial intelligence, and had some ideas about representing information. Upon hearing these ideas, a colleague directed Shapiro to the companies InXight and TheBrain, both of which sell proprietary visualization tools.

Inspired by the possibilities, Shapiro decided to experiment with his own ideas. Having recently taught himself Java, he created TouchGraph and released it as open source in order to evaluate interest. "There wasn't much [interest] at first," says Shapiro, "but it was enough for me to keep at it." In January 2001, Shapiro left his job at Sapient to found his own company and work as an independent consultant. In his spare time, Shapiro further developed TouchGraph, and by May 2002, he was working on his tool full-time.

Quantifying TouchGraph's users is challenging, because of the niche it occupies—user interface components. TouchGraph itself is not an end user application. TouchGraph's users are software developers, who incorporate it into their own applications. However, because TouchGraph's purpose is to present information to end users in a useful manner, nearly all applications incorporating TouchGraph will target end users, and TouchGraph is likely to be the most visible component in those applications. For example, the TouchGraph GoogleBrowser, is a stand-alone application targeting end users, and it clearly showcases the TouchGraph technology. In order to truly quantify TouchGraph's user community, one would have to include both the software developers who use TouchGraph, and the end users who use applications written by these developers.

Estimating the number of end users using applications based on TouchGraph is difficult, although the project's Web site indicates significant attention from both the press and various blogs. Applications that use TouchGraph include a Topic Map viewer and various ontology editors. TouchGraph-based applications have been used to depict computer networks, organize topics on a television show, show relationships between job offers, and display database table structures.

Estimating the number of TouchGraph users—the software developers integrating the component into their programs—is a bit easier. The news section on the TouchGraph web site lists about 20 applications based on TouchGraph, all of them written by different people. TouchGraph is hosted at SourceForge, and according to Shapiro, there are about 20 people subscribed to receive announcements of new releases.

For Shapiro, TouchGraph is an ongoing experiment, one that has slowly but steadily progressed. He says, "My initial goal was to gauge the interest in the dynamic graph navigation concept, and to popularize the concept, and I believe that having the code as open source helped me to do that."

3.1.1. Community

On the surface, TouchGraph is the classic lone developer—or "cave"—open source project (Krishnamurthy). Shapiro works on TouchGraph full-time, and retains complete control over the development and release process. While he is the only person who has write access to TouchGraph's CVS repository, several outsiders have contributed indirectly to the TouchGraph code.[2]

---

[1] See *TouchGraph News* for references to some of these articles.

[2] CVS ("Concurrent Versions System", http://www.cvshome.org/) is a widely used, open source source code control system. It allows multiple people to contribute to a common set of documents, while tracking the different versions of those documents. Many open source projects allow anyone to read from their CVS repository, but restrict write access to their most active developers.

Shapiro says, "The best use for outsiders' contributions has been to learn from their example, rather then applying the changes directly." For example, Shapiro credits Murray Altheim, a graduate student at UK's Open University and a former employee at Sun Microsystems, for helping Shapiro conform to Java coding standards. Shapiro also cites Martin Spermau, Alf Eatons, and Christian Langreiter as programmers who have had a significant impact on TouchGraph's development.

Many of his active contributors are based in Europe. For example, of the four developers cited above, Altheim is currently in the UK, Spermau and Langreiter are in Germany, and Eatons is based in the U.S. Shapiro credits this trend to Europeans embracing the Semantic Web, which emphasizes the modeling the relationships of information, more so than Americans.

Shapiro estimates one to two people actively contributing to TouchGraph at any given time. He has not considered giving any of these contributors direct access to the CVS repository. "Honestly, giving people access to the CVS repository would create too much chaos for me to manage. I am more a fan of the idea of a modular approach, where every user has control over a particular sub domain. Unfortunately, TouchGraph's code is not yet modular enough to let people make upgrades to independent bits and pieces."

Shapiro interacts with his community over private e-mail and the SourceForge forums. The forums seem to have evolved into a user support mechanism, whereas the majority of the active contributions occur over private e-mail. For example, none of the contributors mentioned above have ever posted to the forums. According to Shapiro, "The most useful aspect of the [forums] has been for people to contribute ideas. There have been some nice technical suggestions as well, but it's mostly people complaining that they don't understand the code. Luckily, the forums have in part matured to the point where developers answer each others' questions."

3.2. SquirrelMail

SquirrelMail is a Web-based e-mail client that allows users to read and write e-mail from any Web browser. Luke and Nathan Ehresman had both recently graduated from high school when they started the SquirrelMail project on November 18, 1999 (Ehresman). Their goal was to develop a small, fast, easy-to-use Web-based IMAP e-mail client written entirely in the PHP programming language.

SquirrelMail was one of the first projects hosted on the SourceForge open source software repository site, and it has consistently ranked among SourceForge's most active projects (SourceForge). The project claims to have at least two million users (Castello, Koerkamp). At least one major ISP (Netherlands-based XS4ALL, which has about 100,000 members) and several universities have installed SquirrelMail for members of their communities (Castello, Angliss).

SquirrelMail supports over 30 different languages, including French, German, Spanish, Korean, Japanese, and Icelandic. It also supports plug-ins, which are additional program modules that extend the functionality of the main application. For example, there are plug-ins for mail and spam filtering, for advanced address book features, and for sending automatic responses to messages while on vacation.

While SquirrelMail is an end-user application, its users are not necessarily expected to install or administer it themselves. Typically, a network administrator with access to a Web server installs and maintains the software. Once the software is installed, end-users may run SquirrelMail by accessing a URL via their Web browsers, just as they would use a commercial Web-based e-mail application, such as Hotmail or Yahoo! Mail.

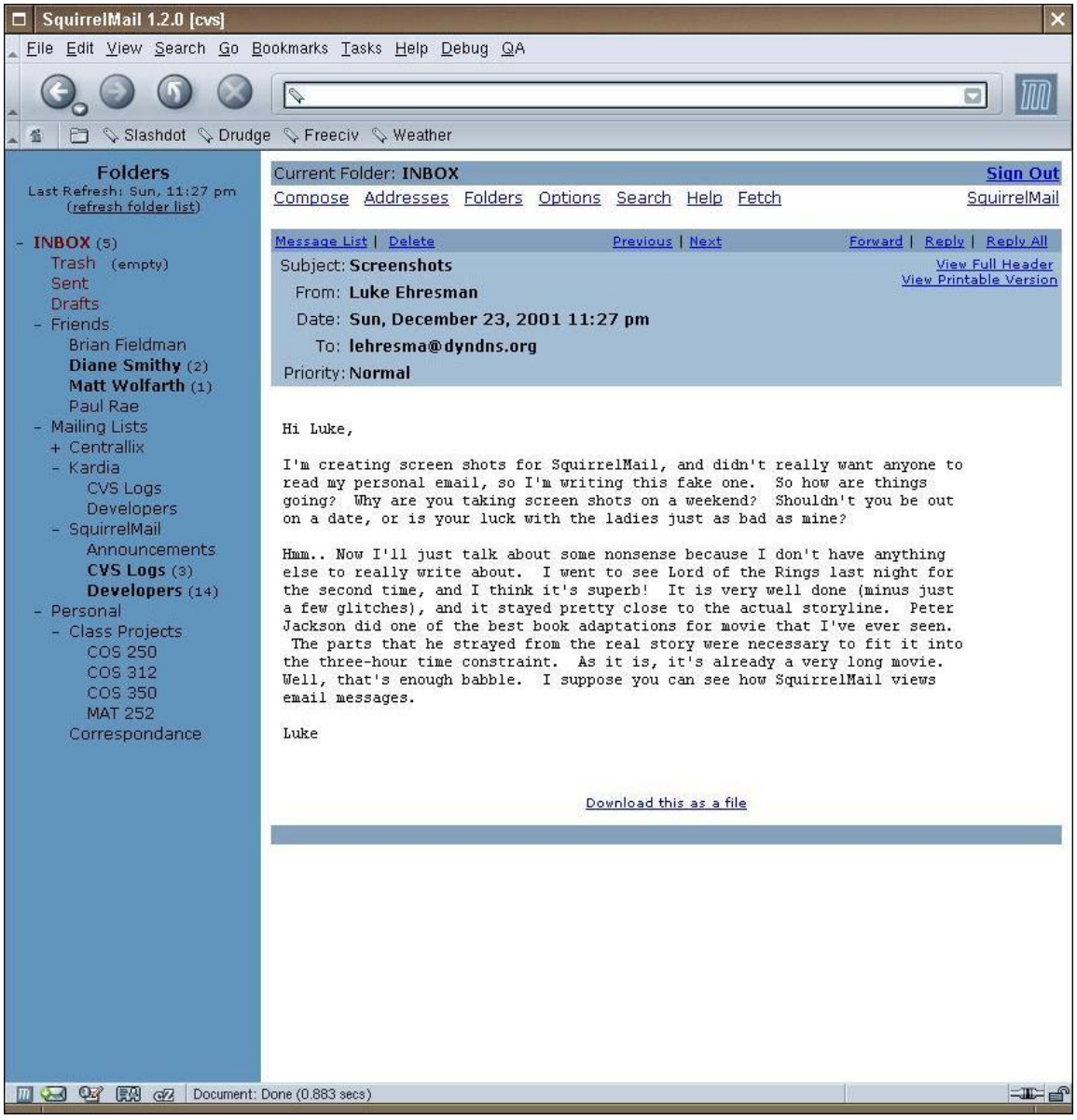

Figure 3.2. Reading e-mail using SquirrelMail (http://www.squirrelmail.org/images/shots/1.2.0/read.jpg).

3.2.1. Community

Most people consider SquirrelMail's "core team" to consist of its 10 project leaders along with two to five other active contributors. SquirrelMail divides its activities into seven projects: stable release, development release, internationalization, plug-ins, user support, documentation and evangelism, and system administration. Rick Castello, a 29-year old Massachusetts-based IT consultant and retail business owner, is the overall project lead.

Of the 12-15 core contributors, all of them except one are male. Most of them are in their mid to late 20s. The youngest is 21, and the oldest is 37 (SourceForge). None of them are paid directly to work on SquirrelMail, although some have been paid to do so in the past, and some who work as network administrators consider contributing to SquirrelMail indirectly part of their job responsibilities.

Over half of the project leads hail from Europe (Spain, Netherlands, and England), while the rest are located in Massachusetts, Texas, and New York. Other states represented among active contributors are Kansas and Virginia (Castello).

Nevertheless, many people who were active earlier in the project continue to be visible in some capacity. For example, all three of the former leaders (Luke Ehresman, Peter Hutnick, and Paul Thompson) continue to post to the development list, although their levels of activity vary. Several of the former project leads continue to be active contributors as well.

The vast majority of the interaction occurs over mailing lists and IRC. In addition to several public mailing lists, there is a private mailing list for project leads. Many of the project leads lurk on the IRC channel all day, every day, ready to answer questions or participate in discussions as they arise (Munro).

The majority of the core team has never met each other face-to-face. Nevertheless, several core team members have said that they consider the others friends as well as colleagues (Angliss, Munro, Castello). Team members talk about their lives and their families over IRC, and they also communicate over the telephone, instant messaging, and private e-mail. Four active members of the team recently started blogs that all link to each other.

Jason Munro, formerly the project lead of the stable release, said that a supportive community and an overall low barrier to entry are important values of SquirrelMail's community, citing his own introduction to the project as an example. Munro dropped out of college in 1991, and after a few years, became interested in computers. He taught himself Linux as well as some C++ and Perl. In 2001, Munro got a job as a network administrator at Standard Beverage Company, a Kansas-based liquor distribution company. One of his first tasks was to find an e-mail package to replace the company's Novell e-mail system, which supported about 120 users spread throughout Kansas.

Several developers tracked all new changes to the source code, providing feedback to other developers if they saw problems. In particular, Mingo seemed to review Munro's changes closely, constantly offering feedback on how to improve his code.

Despite the apparent openness and camaraderie within the group, the SquirrelMail community has had its difficulties. Castello, the project's leader, remarked that some of the community's past difficulties seemed to have stemmed from cultural differences in communication between the Europeans and Americans. Marc Groot Koerkamp, the co-lead of new development and a Netherlands resident, said, "American people cannot always deal with the direct communication we use in the Netherlands. Currently, it's not a problem but we had some tough discussions that didn't go very well."

Castello agreed with this assessment, and added that for many Europeans, English is their second or third language, and what they say is often harsher than what they mean. Castello was also quick to add that major problems did not occur often, and those that did were all eventually resolved.

3.2.2. Process

SquirrelMail's development process has evolved significantly over the past three years. Luke Ehresman was the original leader of the project, and while the community grew fairly quickly, there were no formal roles. After a few years, Ehresman decided to step down as leader of the project to focus on his schoolwork, and Peter Hutnick took over.

One of Hutnick's first moves was to organize an ad-hoc "steering committee," consisting of the most active members of the community (Castello). Out of those discussions, some formal roles were created, including a stable release manager, who would decide when to implement a feature freeze and focus on stabilizing the code for release.

Paul Thompson succeeded Hutnick, and under his leadership, several other roles were formalized. Each lead was responsible for making decisions that affected their subproject, although Thompson retained veto power over all decisions.

One of the most important changes was the division of the project into a stable and development release. New features would be developed for the development release, and when the code matured, the project would create a new stable release. Only bug fixes went into the stable release (Munro).

When Thompson decided to step down, Castello took over. One major change under his leadership has been assigning two leads to each project rather than just one (Castello). This was a reaction to project leads—all of whom were volunteers—often burning out from the responsibilities and stresses of their positions.

All of the decisions must be approved by all of the project's administrators, with the overall leader reserving some veto powers. However, there has never been a formal vote within the community; most of the decisions have been the result of consensus reached via discussion.

Project leads have generally chosen their successors, sometimes after discussions with the other leads. In some cases, people lobbied for their positions, whereas in other cases, the administrators proactively encouraged people to become leaders (Castello). Although most of the role assignments have not been controversial, there have been exceptional cases. Those issues have been resolved to most people's satisfaction without resorting to extreme organizational tactics, such as voting to expel people from the community. Much of this effort occurs in private, via IRC, personal e-mails, and telephone (Castello).

3.3. Discussion Statistics

The descriptions above demonstrate that both TouchGraph and SquirrelMail are compelling applications with active, although markedly different, open source communities. TouchGraph is an innovative user interface component occupying a very small niche. Alex Shapiro, the project's leader, does all of the development himself, with questions and suggestions contributed by others. The user base is small and fleeting, but also consistent, according to Shapiro.

SquirrelMail, on the other hand, occupies a well-established niche—e-mail—and is a widely-used open source application with a core team of very active contributors who drive the project's development. There is a commonly understood community process for developing, releasing, and supporting the software, and that process continues to evolve. Its community is significantly larger than TouchGraph's.

One characteristic that both projects share is that online interaction between both developers and users plays a crucial role in the success of these projects. The purpose of this section is to determine whether there are meaningful quantitative metrics of community interaction for the TouchGraph and SquirrelMail projects. This section presents a number of statistical measures of the online discussion archives from both projects, examines them in light of the projects' qualitative descriptions presented above, and then compares the two sets of metrics in order to identify meaningful trends. Determining whether these trends apply to other open source projects should be a rich avenue for future research.

	Time Span	Number of Posts	Number of Posters	Posts/Person		Mean posts/ month	Mean posters /month	Mean new posters/ month	Number of posters responsible for 50% of posts
				Mean	Median				
Touchgraph	July 2001 - March 2003	129	38	3.4	2	6.14	3.71	1.81	4
SquirrelMail	April 2002 – March 2003	2206	259	8.5	2	183.42	58.25	21.58	14

	Time Span	Number of threads	Thread Depth		Messages/thread		Posters/thread	
			Mean	Median	Mean	Median	Mean	Median
Touchgraph	July 2001 - March 2003	51	1.2	1	2.53	2	1.9	2
SquirrelMail	April 2002 – March 2003	868	1.15	0.5	2.54	2	1.96	2

Table 3.1. Online discussion forum post and thread analysis.

3.3.1. TouchGraph

The content of the touchgraph.development SourceForge forum spans from July 2001 to March 2003. There were 129 total posts contributed by 38 different people.[1] The mean was 3.4 posts/person; the median was 2 posts/person.

Four people (out of 38) were responsible for 50 percent of the posts. Project lead Alex Shapiro topped the list, with 29 percent of the posts. As noted earlier, none of the people Alex cited as active contributors (besides him) have ever posted to the forum. At the same time, the forums consistently attracted 1.81 new people every month. These numbers are consistent with Shapiro's observation that one or two people are active at any time period. There were an average of 6.14 posts/month, and 3.71 unique posters/month.

There were 51 total threads of discussions. The mean depth of each thread was 1.2 levels, and the median was 1.[2] The mean number of messages per thread was 2.53, and the median was 2. The mean number of posters per thread was 1.9, with a median of 2.

3.3.2. SquirrelMail

SquirrelMail's main forum for developer interaction is the squirrelmail-devel mailing list. From April 2002 to March 2003, there were 2,206 total posts submitted by 259 different people. The mean posts/person was 8.5, with a median of 2. 14 people were responsible for 50 percent of the messages posted.

It's immediately evident that SquirrelMail is significantly more active than TouchGraph, with a larger number of active posters. Of the 14 most active posters, 10 (71 percent) are current or past members of the core team (as listed on SquirrelMail's web site). Of those 10, five are current project leads, and three are former project leads. In other words, five of the 10 current project leads are among the most active participants on the squirrelmail-devel mailing list. On average, there were 183.42 posts/month and 58.25 unique posters/month. 21.58 of those 58.25 posters each month were new.

The SquirrelMail mailing list has 32 times the amount of traffic as TouchGraph's forum. However, despite the order of magnitude difference in traffic, squirrelmail-devel's ratio of new monthly posters to total monthly posters is equivalent to TouchGraph's — approximately 30 percent. In other words, 30 percent of the people posting on both SquirrelMail's and TouchGraph's forums each month are people who have never posted to those forums before. One possible hypothesis about successful open source communities is that a steady stream of new contributors is necessary to maintain a project's momentum. A potential avenue of future research might be to see if this 30 percent ratio is found in other successful open source projects as well.

There were 868 total threads of discussion. The mean depth was 1.15 levels, and the median was 0.5. The mean number of messages per thread was 2.54, with a median of 2. The mean number of posters per thread was 1.96, with a median of 2.

Despite the order of magnitude difference in traffic and people posting, the average depth, number of messages, and number of participants in each thread are almost identical for SquirrelMail and TouchGraph. This suggests that, despite the size differential between the two communities, there may be strong similarities in the types of interaction that occur on their main forums.

---

[1] In this study, I counted every unique e-mail address as an individual participant. However, an individual may use more than one e-mail address. For more accurate results, these e-mail addresses should be aggregated.

[2] The depth of a thread equals the number of levels of responses. For example, a thread with a single message has a depth of zero. A thread with one or more messages responding to a message has a depth of one. A thread with a message responding to a message responding to a message has a depth of two.

The low averages for the threads on both forums imply that the majority of the discussions are not complex, and reach resolution quickly. Both communities carry out much of their design and organizational discussion privately. If those discussions were on the public forums, the means would probably be higher.

3.4. Patterns of Collaboration

Full comprehension of open source software communities requires a more complete understanding of the software development process than provided here. However, there are patterns underlying these organizational processes that are not specific to software development. These patterns can be found in other successful communities as well. One of the main goals of this research is to identify these patterns, and to develop a pattern language that can be used to describe, build, and improve other types of successful communities.[1]

The following are patterns observed in both the TouchGraph and SquirrelMail communities.

3.4.1. Evolve the Community

It is extremely difficult to predict what kind of interest an open source project will attract. Designing an organizational structure for what might be, rather than what is will likely impede the project rather than facilitate it.

SquirrelMail's organizational structure and processes emerged over time. Its system of subprojects and project leads worked because the code was modularized, and there was already an active community of participants from whom to draw. Had Luke Ehresman, SquirrelMail's founder, tried to impose this structure when he first started the project, it likely would have failed, because the necessary roles would not have been clear at that point, and there were no candidates to whom to assign those roles.

Alex Shapiro, TouchGraph's creator, has not delegated CVS commit access to members of his communities, because he doesn't see the need, and he doesn't think his code is modular enough. He also recognizes that doing so would create unnecessary organizational overhead with no immediate benefits.

Both projects have been reactive rather than proactive. They have allowed an organizational scheme to emerge, rather than attempting to impose one.

3.4.2. Lead by Example

To celebrate the one year anniversary of SquirrelMail, Ehresman wrote an essay describing the lessons he had learned. He noted that the more active the leader is, the more active the community will become. "A strong correlation exists between developer activity and my personal excitement and involvement in the project," Ehresman said. "Whenever I took a week or two off, not much development happened—on the flip side of that, when I was ecstatic about certain aspects of the project, developer response and activity was quite high. It is important that your developers see your enthusiasm so they can share in your excitement."

As a corollary, Ehresman noted that participating on the project's public forums can have an important effect on a community. He said, "Being involved in the mailing lists is a never-ending job, but involvement as project leader is necessary! It helps a lot for users to see active involvement just as it's important for developers to see this"

The leaders of TouchGraph and SquirrelMail are both active and visible within their communities. Shapiro updates the News section of TouchGraph's Web site periodically, and he is responsive on the touchgraph.development forum. Similarly, 10 of the top 14 participants on the squirrelmail-devel mailing list are members of the project's core team.

---

[1] The notion of a pattern language is borrowed from the architect Christopher Alexander, who introduced the concept in his book, *Timeless Way of Building*.

Leading by example is especially crucial for open source communities, because its participants are largely volunteers. You cannot simply delegate a task to participants and expect them to do it if you have not first earned your authority. If you yourself are not working hard, enthusiastically, and visibly, then you will not attract others who will.

3.4.3. Users Talk to Developers

With both TouchGraph and SquirrelMail, users and developers are part of the same community. They interact on the same public forums, and in both cases, many users become active members of the community by answering other users' questions.

Not only are the communication barriers between users and developers small, the barrier for a user to become a developer is small. TouchGraph's most significant outside contributors were all TouchGraph users who found ways to improve TouchGraph's code. Several of SquirrelMail's current project leads, including overall lead Castello, initially joined the community as users, not developers.

4. Conclusions

A great amount is already known about open source communities and its participants. They consist largely of men in their 20s and 30s. The vast majority of them are IT professionals or students residing in the United States and Europe. Most participate in order to expand and share their knowledge. The software development process tends to be controlled by individuals or small teams of developers.

Additionally, the case studies presented in this report hint at how collaboration works within successful projects. Project leaders tend to be active on public forums, but also collaborate on private forums. The public forums attract a steady stream of new participants, a stream that is necessary to balance out a relatively high rate of attrition. Community processes are lightweight, and tend to emerge in response to changing conditions.

Nevertheless, there are still many aspects of open source communities that are not well understood, and are worthy of further study.

- **Metrics.** Analyzing discourse is vital to understanding collaboration. One of the advantages of online communities is that the majority of this discourse is archived digitally. As a result, some aspects can be measured automatically. This report looked at several aspects of online discourse, and identified a few possibilities for measuring community effectiveness. For example, one way to measure the openness of a community is to determine how often new participants post on public forums. In both the TouchGraph and SquirrelMail communities, about 30 percent of people posting on the public forums each month were first-time participants. It would be valuable to see if there are similar, measurable trends in other successful open source communities.

- **Coevolution.**[1] There are several well-defined roles within software development communities: software architects, programmers, release managers, testers, etc. What is not yet well understood are the roles that users play in the software development process. This question is even more important in the context of open source communities, where the barriers between developers and users are generally quite low. What seems clear from the examples above is that users can play a much more significant role than simply reporting bugs or evangelizing the projects. An important research goal is to identify what those roles are, and how they affect the overall software development process and community dynamic.

- **Models of Knowledge Sharing.** Surveys show that the vast majority of open source participants join projects in order to acquire and share useful knowledge. One obvious question is, what techniques do open source communities use to manage knowledge? A related question concerns how knowledge moves from person to person. John Seeley Brown and Paul Duguid have proposed that the Silicon Valley is particularly conducive to innovation because there are networks of knowledge transfer that transcend organizational boundaries (31-32). Open source communities seem to exhibit similar traits. One common occurrence worthy of detailed examination is "forking": small groups of people forming new communities around already existing open source projects, and leading those projects in new directions. How do the macro processes of the open source community as a whole affect the processes within individual projects?

If the TouchGraph and SquirrelMail case studies are any indication, open source communities promise to be a fertile source of reusable patterns of high-performance collaboration. Continued research in these areas would not only result in a better understanding of open source communities, it would facilitate the development of a strategy for improving all types of communities.

[1] "Coevolution" is a term coined by Doug Engelbart to describe how tools and their users symbiotically influence each other's evolution (Engelbart).

5. Works Cited

Alexander, Christopher. *Timeless Way of Building*. New York: Oxford University Press, 1979.

Angliss, Jonathan. E-mail interview. 17-20 March 2003.

Brown, John Seely and Paul Duguid. "Mysteries of the Region: Knowledge Dynamics in Silicon Valley." *The Silicon Valley Edge*. Ed. Chong-Moon Lee, William F. Miller, Marguerite Gong Hancock, and Henry S. Rowen. Stanford, CA: Stanford University Press, 2000.

The Boston Consulting Group Hacker Survey. Lakhani, Karim R., Bob Wolf, Jeff Bates, and Chris DiBona. Release 0.73. 24 July 2002 <http://www.osdn.com/bcg/BCGHACKERSURVEY-0.73.pdf>.

Castello, Rick. Telephone interview. 21 March 2003.

---. Telephone interview. 22 March 2003.

Ehresman, Luke. "A Year of Learning." 27 January 2001 <http://www.luke.ehresman.org/articles/comp/learning.php>.

Engelbart, Douglas C. "Toward High-Performance Organizations: A Strategic Role for Groupware." Bootstrap Institute. June 1992 <http://www.bootstrap.org/augment/AUGMENT/132811.html>.

Free/Libre and Open Source Software: Survey and Study Final Report. International Institute of Infonomics, University of Maastricht, The Netherlands and Berlecon Research GmbH, Berlin, Germany. June 2002 <http://www.infonomics.nl/FLOSS/report/>.

Judge, Peter. "Ballmer: United, we'll stomp on Linux." *CNET News.com*. 24 September 2002 <http://news.com.com/2100-1001-959165.html>.

Koerkamp, Marc Groot. E-mail interview. 20 March 2003.

Krishnamurthy, Sandeep. "Cave or Community: An Empirical Examination of 100 Mature Open Source Projects." *First Monday* 7.6 (June 2002). <http://www.firstmonday.dk/issues/issue7_6/krishnamurthy/>.

McGovern, Pat. "SourceForge Sitewide update." E-mail to SourceForge members. 18 March 2003.

Munro, Jason. Telephone interview. 21 March 2003.

The Open Source Definition. Open Source Initiative. <http://www.opensource.org/docs/definition.php>

Shapiro, Alex. E-mail interview. 11 February 2003 – 14 March 2003.

SourceForge.net: Project of the Month, January 2003. SourceForge. January 2003 <http://sourceforge.net/pom_0103.php>.

TouchGraph News. TouchGraph LLC. <http://www.touchgraph.com/news.html>.

Two Case Studies of Open Source Software Development: Apache and Mozilla. Mockus, Audris, Roy T. Fielding, and James D. Herbleb. Avaya Labs Research. March 2002 <http://www.research.avayalabs.com/techreport/ALR-2002-003-paper.pdf>.

Watkins, Don. E-mail correspondence. 21 March 2003.

WIDI – Who Is Doing It? 14 August 2001. Technical University of Berlin. <http://widi.berlios.de/>.