An Examination of Initiation, Organization, Participation, Leadership, and Control of Successful Open Source Software Development Projects

Michael P. Conlon
michael.conlon@sru.edu
Computer Science Department,
Slippery Rock University of Pennsylvania
Slippery Rock, Pennsylvania, U.S.A.

Abstract

While there have been many studies of software development projects, the fact that most open-source software development projects are managed in a very different way from proprietary projects may make the lessons from such studies inapplicable to open-source development. This paper examines several successful open-source software projects, in an attempt to elucidate how successful open-source projects are managed, and what facets of their initiation, leadership, organization, and control have contributed to their success.

Keywords: systems analysis, open source software, free software, software development, software engineering

1. INTRODUCTION

Motivation

There are many open-source software development projects. This paper attempts to determine, from publicly-available information, how several successful open-source projects were initiated, and are led and controlled, and the extent of participation in the programming portion of these projects.

Open-source Software

Because of the common confusion among open-source, free, public-domain, and shareware software, it is customary to begin discussions about open-source software with an explanation of what open-source software is, its history, and what distinguishes it from other types of software. This paper will follow that custom.

Open-source software is sometimes called *free software* when the speaker or writer wishes to emphasize the freedoms conveyed to the user of the software (Free Software Foundation, 2005):

- The freedom to run a program, for any purpose.
- The freedom to study how the program works, and adapt it to your needs.
- The freedom to distribute copies.
- The freedom to improve the program and release your improvements to the public, so that the whole community benefits.

There is no essential difference between *free software* and *open-source* software: such software is supplied with its source code so that users can modify it to meet their needs, and is generally available free of charge.

Note, however, that open-source software *is* often sold, for open-source licenses do not prohibit copying and resale of the software. Often when open-source software is sold, some level of support is included in the price.

Open-source software should not be confused with *public-domain* software, of which there is little of significant scope or quality, nor with *shareware*, which is merely proprietary software with a try-it-before-you-buy-it license.

A particularly popular and interesting free software license is the *General Public License (GPL).* The GPL is often called a *copyleft* license, because it reserves the above freedoms for the user, thus using copyright law to preserve the right of users to, among other things, copy. An important feature of the GPL is that it requires that those who distribute modified versions of the software must use the GPL as the license for the modified versions. This prohibits the turning of GPL'd programs into proprietary programs. Other open-source licenses, such as the Berkeley (BSD, or Berkeley Software Distribution) license, do not have this "viral" feature.

Roots of Open Source Software

Open Source software originated in the user groups of the major computer hardware vendors and in the computer science laboratories of universities, where a culture of sharing software prospered. However, until recently there was little open-source software for Microsoft's Windows operating system. Open-source development occurred mostly in the Unix community. This situation has gradually changed, largely because programmers developing for Unix/Linux have begun using cross-platform GUI libraries. There are now many good open-source programs available for Windows systems. All of the open-source programs discussed in this paper, with the exception of the Linux operating system, are available in versions that will run on the Windows platform.

Forking

Essential to the open source/free software community is the right to fork. Forking refers to a second group of developers taking the source code of an existing project and diverging the development into a different code base. While forking is sometimes portrayed as a problem, it is an essential freedom and protection, as explained below.

A fork occurs when some individuals disagree with the direction a project is taking. They then recruit others to start developing the project in a different direction. When a fork occurs, the rest of the community must decide which version of the project to follow. Normally, the vast majority of the community will make the same choice, and one "tine" of the fork will thrive and the other will whither. The result is that the community is better served, for no individual or team can maintain complete control over, or *hijack*, the project code. Note that, in closed-source development where the code is owned by a single corporation, a fork never occurs and the community has no such veto over the direction of development.

A classic example of a successful fork is the 2004 creation of the *X.org* project from the *XFree86* project. According to (Wikipedia, 2007),

> *In February 2004, with version 4.4.0, the XFree86 Project adopted a license change that the* Free Software Foundation *considered* GPL *incompatible.* [This means that XFree86 code could not legally be mixed with GPL'd code. This was a serious problem because most open-source code is licensed under the GPL.] *Most* Linux distributions [A distribution is the operating system combined with a collection of application and utility programs to comprise a useful system.] *found the potential GPL legal issues unacceptable and made plans to move to a fork from before the license change. At first there were multiple forks, but the* X.Org fork *soon took over as the dominant one. Most of the developers who were already annoyed at other issues in the project also moved to X.org.*

With the X.org fork, license compatibility with the GPL was maintained, and development of X.org accelerated substantially compared to the development pace of XFree86. Thus, the fork was a double "win" for the community of users and developers of X.org. The principal concepts here are that open source code makes

forking possible, and that any open source project that ignores the best interests of its development and user community risks a fork.

2. OVERVIEW

Successful Open-source Development Projects

We will examine several open-source projects, in order to make the salient features of each project clear. These projects are:

- The Linux Kernel, which is the highest-profile open-source project. Linux proved that loosely-controlled, distributed design and development could be effective.
- Apache, the dominant Web server. Apache is often mentioned with Linux because of its comparable history and success.
- MySQL is a relational database management system that was initiated in spite of entrenched competition. It now challenges even enterprise database management systems such as Oracle.
- PostgreSQL is one of the oldest open-source projects. It is an object-relational database management system.
- OpenOffice.org, a suite of office applications: word processor, spreadsheet, presentation program, etc. Originally a proprietary office suite called *StarOffice*, it was purchased by Sun Microsystems, which removed portions of the code it didn't own, and open-sourced the rest, calling it OpenOffice.org. (StarOffice is still available from Sun as a proprietary product.)
- Alice. Originally a virtual-reality program generator, Alice has morphed into a highly-motivational system for introducing beginners to computer programming by animating existing three-dimensional cartoon objects.

While there are many similarities among these projects, there are important differences among them as well. In order to learn the lessons from each of these projects, we need to answer the following questions:

- How did the project get started?
- How and why did it acquire a community of developers, and how large is this community?
- How open is the project to new developers?
- How is project leadership organized?
- How is control exerted over the development process?

3. THE LINUX KERNEL

Initiation

The Linux project started when Linus Torvalds, then a graduate computer science student at the University of Helsinki, began looking for an operating system better than DOS/Windows for his laptop computer. He initially tried Andrew Tanenbaum's *Minix*, but soon grew frustrated with the limitations of Minix as well. He started developing his own Unix-like operating system, then posted a message on the Minix Usenet news group in 1991 inviting programmers to try it and contribute to it.

Community of Kernel Developers

Linux quickly attracted a community of developers with interests and frustrations similar to Torvalds'. There were apparently many programmers who had used Unix at work or at their college or university who wanted it, or, in this case, a clone, on their PC's, and who had the time and talent to help create it.

Version 2.6.0 of the Linux kernel provides us with a snapshot of the Linux development community. This version was in development for 680 days and accumulated 27,149 changes over its predecessor. This results in an average of 1.66 changes per hour during the period. There were 916 developers who contributed at least one change to the code (Kroah-Hartman, 2004).

In a study of commits (code additions or modifications) to the Linux kernel source code tree over the period April, 2005 to November, 2006, (Altonen and Jokinen, 2006) found that 1722 individuals had contributed.

Whether the true size of the Linux development community is closer to 916 or to 1722, it is apparent that it is quite large. There is probably no other software project, open or closed, with as large an active development team.

Linux Leadership and Control

There is no formal organization that controls the Linux source code or directs and finances development. However, many, if not most, major contributors are employed by computer companies such as IBM, Hewlett-Packard, or Red Hat, and perform Linux development as their primary responsibility to their employers.

Linux development community is organized as a tree. Linus Torvalds is the chief developer. He has a small number of lieutenants, each of whom is responsible for a major subsystem of Linux. Other developers submit their code to a lieutenant or sub-lieutenant, who evaluates it and, perhaps, inserts it into his subsystem. Periodically, subsystems are passed up the tree to the next higher developer who adds it to the official code base at his discretion. [*His* is used advisedly. There are virtually no women involved in open-source programming.]

This type of organization is sometimes called the *benevolent dictator* organization. No code enters the repository unless approved by a lieutenant and by Linus himself. Linus has earned his rank in the project by initiating the project, by his acknowledged technical skills, and by his interpersonal and public-relations skills.

It has often been said that one of Torvalds' greatest contributions was his practice of posting updated versions on the Internet *daily*, and sometimes several times daily (along with warnings about the untested nature of the new code). This kept his volunteer programmers interested and attracted new ones, and they rapidly found the bugs, fixed bugs, and wrote new code for Linux.

There is no apparent attempt by Linus or his lieutenants to dictate the direction of development. Each contributor works on a subsystem of his own, or his employer's own, interest. However, any contributions must be deemed as worthwhile and competent additions to the kernel by the lieutenants and by Linus, or they will be rejected.

There are portions of the kernel that are generally acknowledged to need improvement; e.g., recently there has been significant discussion about the need for a better process scheduler. Nonetheless, leadership does not assign work to developers: developers must step forward and volunteer to do work that needs to be done.

This is true in a more-general sense: all Linux development is performed by volunteers, and anyone can participate by writing some code. However, technical criticism of poorly conceived changes can be brutal, and only talented designers get promoted up the tree of control.

4. APACHE

Initiation

Apache began its life as *httpd,* which remains its filename. *Httpd* was developed by Rob McCool at the National Center for Supercomputing Applications at the University of Illinois at Urbana/Champaign. However, McCool left NCSA in 1994, and httpd development stalled. It was left to the webmasters who used httpd to maintain it for themselves. Soon, several of them, led by Brian Behlendorf and Cliff Skolnic, organized themselves for their mutual benefit. A central archive was set up, and the webmasters submitted patches to the archive. The resulting Web server was renamed *Apache*. (The Apache Software Foundation disputes the claim that the name came from the fact that they had developed "a patchy server.") Eventually, Apache was rewritten from scratch, and, today, contains

little or no NCSA code. (The Apache Software Foundation-1, 1999-2005)

Apache's Developer Community

Because NCSA httpd was one of the first Web servers, it had attracted a considerable clientèle of webmasters. Because these were the early days of the Web, the typical webmaster was both a programmer and a system administrator, and accordingly possessed considerable design talent. When NCSA stopped supporting httpd, these webmasters had nowhere to go for support but to each other. Thus, Apache inherited a critical mass of talented developers. Note that it was, initially, httpd that attracted them. Apache inherited them, and subsequently attracted its own developers.

The Apache Software Foundation, which oversees several projects besides just the Apache Web server, claims "almost 800" committers (The Apache Software Foundation-3, 2007).

Apache Leadership and Control

Apache development is controlled by a formal organization called the Apache Software Foundation. There are many Apache projects, but most of them are related in some way to their flagship Web server. According to their Web site,

> *The Apache projects are characterized by a collaborative, consensus based development process, an open and pragmatic software license, and a desire to create high quality software that leads the way in its field. We consider ourselves not simply a group of projects sharing a server, but rather a community of developers and users.* (Apache Software Foundation-2, 2007)

Each project is led by a Project Management Committee, and membership on the PMC is limited to committers. I.e., the team is a meritocracy where those who have written substantial parts of the project govern the project. Among its other responsibilities, each PMC is responsible to see that each program is the result of a technical consensus, and not that of a small clique. Interested individuals can work their way up the Apache project hierarchy first by being a user of the software, then becoming a developer (contributing code), eventually becoming a committer (one with write-access to the code base), and then joining the PMC for the project (Apache Software Foundation-3, 2007). Of course, this presupposes that the individual has the competence to do each of these jobs well.

5. MYSQL

Initiation

Unlike the Linux kernel and Apache, MySQL is the property of a commercial enterprise, MySQL AB. According to the MySQL AB Web site,

> *The company was founded in Sweden by two Swedes and a Finn: David Axmark, Allan Larsson and Michael "Monty" Widenius who have worked together since the 80's. MySQL AB is the sole owner of the MySQL server source code, the MySQL trademark and the mysql.com domain worldwide.* (MySQL AB-1, 2007)

Thus, while MySQL is an open-source project, it is at the same time a proprietary project. According to the MySQL "Contributor License Agreement," code contributions from the MySQL community become the property of MySQL AB (MySQL AB-2, 2007).

MySQL began as a project of Monty Widenius in 1995, as a part of a data warehousing system.

MySQL's Developer Community

With the explosion of the Worldwide Web in 1995, MySQL quickly attracted the attention of webmasters seeking an open-source database. As with other open-source projects, users soon offered debugging assistance and wrote code to enhance the product.

One challenge MySQL AB finessed was that of growing a community of developers while keeping control over the MySQL code base. They did this with a dual-licensing model,

and with enticements for contributors to surrender the copyright to their contributions. Users could download and use a "community" version licensed under the General Public License, or they could pay a fee to use MySQL under a proprietary license that allowed them to do things that the GPL prohibits, and to obtain software support. MySQL AB gets its income from license and support fees from users who purchase the proprietary license, and it obtains debugging and development assistance primarily from users of the GPL license.

Such a strategy can work only if MySQL AB provides effective leadership for the project. As long as MySQL AB listens to its community and makes competent development decisions for the improvement of the code, a fork will not succeed. While contributors lose ownership of their code contributions, they gain access to an improved product. Enough people have contributed that MySQL has improved at an impressive rate. Note that a fork would be devastating to MySQL AB's business model: under forked development, MySQL would not receive ownership of bug fixes and upgrades, the fixes and upgrades would be covered exclusively by the GPL, and therefore MySQL AB could not sell these modifications under its proprietary licenses .

Information about the number of active MySQL developers is not readily available, but MySQL AB claims over 11 million active MySQL installations worldwide and over 50,000 MySQL downloads per day (MySQL AB-3, 1995-2007). If .1% of the installations have one person who contributes code, MySQL would have a thousand developers. This would account for the rapid development evident in MySQL.

MySQL Leadership and Control

As previously indicated, MySQL is the property of MySQL AB, a for-profit corporation. While anyone who signs a Contributor License Agreement can contribute to the project, the employees of MySQL AB determine what code gets included. MySQL AB provides their contributors with incentives such as licenses to their enterprise version and public recognition. Nonetheless, there seems to be no formal organization of developers outside the company itself. The organization of the team inside MySQL AB does not seem to be documented publicly.

6. POSTGRESQL

Initiation

Michael Stonebreaker had left the University of California at Berkeley's *Ingres* project. He returned in 1985 to start a post-Ingres project to address perceived shortcomings in the database management systems of the early 1980's. This project was naturally named *Postgres*.

In 1994, Andrew Yu and Jolly Chen at Berkeley added an SQL language interpreter, calling the new version *Postgres95*. Released under the Berkeley open-source license, the code became generally available, but it was orphaned when Yu and Chen moved on.

PostgreSQL's Developer Community

The attraction of PostgreSQL was that it was the first open-source object-relational database. By 1995, Postgres95 had attracted a development community. It was renamed PostgreSQL after substantial re-writing of the code base, including the addition of an interpreter for Structured Query Language (SQL). Today it is maintained by the (PostgreSQL Global Development Group-1, 1996-2007).

Because it is covered by the BSD license, an open-source license which permits use of the code in third-party proprietary products, there have been several attempts at commercializing PostgreSQL. Some of these efforts have had modest success, and several have resulted in significant returns (contributed improvements) to PostgreSQL's open-source versions.

PostgreSQL's Web site lists forty-nine active contributers and seventeen former contributers.

PostgreSQL Leadership and Control

The PostgreSQL project actively encourages code contributions. They have posted specific

instructions for new contributors to follow. They maintain an email address where new code can be deposited. Code submitted to that email address is subsequently reviewed by more-experienced PosgreSQL developers (PostgreSQL Global Development Group-2, 1996-2007). It is not clear how one becomes a code reviewer; presumably in a similar way to that of other open-source projects: by developing a reputation for technical competence.

7. OPENOFFICE.ORG

Initiation

OpenOffice.org began life as *Star Office*, a proprietary product of the German company *Star Division.* (Star Division didn't seem to be a division of anything.) In 1999, Sun Microsystems purchased Star Division, and Star Office became a Sun product. After removing code that belonged to other companies, Sun released the remainder as *Open Office* and as open-source software. Because the term *Open Office* turned out to be an existing trademark of another company, the suite was renamed after its Web site, *OpenOffice.org*.

OpenOffice.org's Developer Community

As the first cross-platform, open-source, complete office suite, with excellent ability to read and write Microsoft Office files, OpenOffice.org attracted a large user base. Sun assigned several programmers to continue the development of OpenOffice.org, and Sun provides the project's Web site. Contributors must sign Sun's Joint Copyright Assignment form, giving Sun joint ownership of the code.

Size of OpenOffice.org Developer Community

There had been rumors that OpenOffice.org has had difficulty attracting a development community outside of Sun employees. However, as of June, 2007, there are one hundred fifty one OpenOffice.org developers who have write-access to the source code. Many others are involved in development with lesser degrees of access (OpenOffice.org Wiki, 2007). It seems unlikely that Sun could afford to assign that many programmers to a program that it gave away free of charge. It should be safe to say that most of them are outside developers. This indicates a strongly-supported project.

OpenOffice.org Leadership and Control

OpenOffice.org leadership and control is remarkably similar to that of Apache. Sun sponsors the project, and CollabNet hosts and helps manage the project. There are members, contributors, developers, and project leads, in ascending order of influence (OpenOffice.org, 2006).

8. ALICE

Initiation

Alice began its life at the University of Virginia under Randy Pausch, where it was developed to automate the development of virtual reality systems. In 1997, Pausch moved to Carnegie Mellon University, bringing the project with him. Somewhere along the way, the purpose of Alice changed. By version 2.0, it had become a system for teaching elementary computer programming to students that had come of age in a graphical computer world.

Alice's Developer Community

Alice has attracted a strong community of users, and source code is available. However, the leaders of the project do not accept bug fixes or enhancements from individuals outside the small project group at Carnegie Mellon. This type of team structure has been called the *cathedral* style, for the designers of the great Gothic cathedrals were few in number and did not permit strangers to walk onto the construction site and make changes to the design (Raymond, 1997). Virtually all proprietary software development follows the cathedral model.

This situation might make Alice ripe for a fork, except that the team is relatively well funded, has recently had the library of *Sims* characters donated to the project free of charge by Electronic Arts, the program is so *revolutionary*, and the team obviously has

the best interest of its users in mind. Until the new version of Alice with the Sims characters is released, a fork is unlikely. Even then, any fork would depend strongly on the wording of the Sims' license.

It seems important to note that all of the other projects described in this paper, and the great majority of open source projects in general, use the contrasting *bazaar* style of structure. In bazaar-style software development, just as in a town bazaar, anyone may come and set up her tent and participate, provided she follows the ground rules of the town or project. (Raymond, 1997)

Alice Development Team Size

The development team seems to consist of fewer than ten programmers.

Alice Leadership and Control

The Alice project is under the aegis of the Stage3 Research Group at Carnegie-Mellon University, which is led by Andrew Pausch who started the project and in whose name external funding has been obtained. Since development is completely internal, there is no information available about how the project is controlled.

It is interesting to note the much-slower pace of development in the Alice project: Version 2.0 was released on April 5, 2005, and has not seen an upgrade or bug-fix release since then (Stage3 Research Group, 1999-2007). It seems that they are concentrating on version 3 with Sims technology. Perhaps the lack of bug-fix releases of Alice can be attributed to the small team size and the lack of outside participation in coding, for these are the two ways that Alice development differs from that of the other projects described in this paper.

9. ANALYSIS

Every one of these projects was started by a single person or small, co-located team who developed interesting, working code. Those that encouraged outside participation used the working code to attract such participation.

Most of the projects have rather large development teams, especially by contrast with typical proprietary-development teams. Few, if any, companies could afford to finance such large teams. The exception here is *Alice*, which does not accept outside contributions and which apparently pays a salary to all or most of its programmers. PostgreSQL has a relatively small development team of 49, but it is also the most mature of the projects and you would not expect it to need major development at this point in its life. It is important to note that, in this paper, only the programmers are being counted. We have not counted testers or documentation writers. We also have not counted managers (who are scarce in open-source development).

A consequence of the large team size is the pace of development. Most of these projects developed quite rapidly, and have a reputation for stability and security. Often, stability comes quite early in the life of an open-source project. From the personal experience of the author, it can be attested that Linux, even as early as V. 0.99, was quite stable and usable. An exception here is Alice, which is a revolutionary tool for introducing programming, but has some of the instability one would expect of an immature project.

Alice contrasts with all of the other projects we have studied in this paper. [See appendix 1.] As indicated previously, it uses a small, closely-controlled development team and does not welcome volunteer developers; it is also the least stable of these projects, and it is the slowest to release bug fixes. The explanation for this may be that the power of the core team is not multiplied by volunteer developers, as it is in almost all other successful open-source projects. While the Alice project welcomes bug reports from its user community, it apparently does not have sufficient programmers to deal with the bugs in a timely manner.

Another characteristic of Alice in which it differs from the other projects is its novelty. Operating systems, database managers, Web servers, and office suites are well-established

technologies. There is only one user-friendly integrated development environment for three-dimensional animation, and that is *Alice*. Andrew Morton, one of the chief developers in the Linux project, has expressed the opinion that open-source development is appropriate only for projects involving established technologies (Morton, 2004). It may be that the apparently slower development of Alice may be due more to its uniqueness and technical challenges than to its cathedral development style. If the entire population of competent developers for a project like Alice is already on the Alice team, not much real help could be expected from outside developers. On the other hand, it is likely that there are many programmers in the world who would rapidly comprehend the intricacies of Alice if given the chance to contribute. It might be worthwhile for the Alice team to allow contributions from at least a small number of outside programmers to see if it would not accelerate development.

10. CONCLUSIONS

If open-source development is being used, the program will improve more rapidly both in features and in stability if the power of volunteer developers can be harnessed. It is not the open source code alone, but the community cooperation that makes open-source development effective.

11. FUTURE WORK

It would be appropriate to compare these six projects with several lower-profile but successful ones, and with several that have been unsuccessful.

12. REFERENCES

All Web references below were accessed and appropriate for the purposes of this paper as of 13 Sept. 2007.

Altonen, Timo and Jokinen, Jyke (2006) "Demography of Linux Kernel Developers." Web document: http://www.cs.tut.fi/~tta/demography.pdf.

The Apache Software Foundation-1 (1999-2005) Apache HTTP Server Project *About* page. Web document: http://httpd.apache.org/ABOUT_APACHE.html

The Apache Software Foundation-2 (2007) Home page. Web document: http://apache.org.

The Apache Software Foundation-3 (2007) "How the ASF Works" page. Web document: http://www.apache.org/foundation/how-it-works.html.

Free Software Foundation (2005) "The Free Software Definition." Web document: www.gnu.org/philosophy/free-sw.html.

Glass, Robert L. (2003) "A Sociopolitical Look at Open Source." *Communications of the ACM*, 46(11), November, 2003, pp. 21-23.

Morton, Andrew (2004) "Open Source Software Development and the Software-using Business World." Transcript of a speech given at SDForum, November 16, 2004. Available at http://www.groklaw.net/articlebasic.php?story=20041122035814276.

MySQL AB-1 (1995-2007) MySQL *About* page. Web Document: http://www.mysql.com/company. .

MySQL AB-2 (2007) "MySQL Contributor License Agreement, " version 0.3. Web document: http://forge.mysql.com/contribute/cla.php.

MySQL AB-3 (1995-2007) "MySQL Fact Sheet." Web document: http://www.mysql.com/company/factsheet.html

Open Source Initiative (2002) The Open Source Definition. WWW document, http://www.opensource.org/docs/definition.php

OpenOffice.org (2006) "Guidelines for Participating in OpenOffice.org." Web document: http://www.openoffice.org/dev_docs/guidelines.html

OpenOffice.org Wiki (2007) Domain

Developer page. Web document, http://wiki.services.openoffice.org/wiki/DomainDeveloper

PostgreSQL Global Development Group-1 (1996-2007), PostgreSQL *History* page. Web document: http://www.postgresql.org/about/history

PostgreSQL Global Development Group-2 (1996-2007), "Developer's FAQ for PostgreSQL." Web document, http://www.postgresql.org/docs/faqs.FAQ_DEV.html

Raymond, Eric (1997) *The Cathedral and the Bazaar*. Web document, www.catb.org/~esr/writings/cathedralbazaar/cathedral-bazaar

Stage3 Research Group, (1999-2007) "Download Alice v2.0." Web document, http://www.alice.org/downloads/authoringtool

Torvalds, Linus (1991) "Free Minix-like Kernel Sources for 386-AT", Usenet News posting. Available at http://groups.google.com/group/comp.os.minix/browse_thread/thread/e3df794a2bce97da/2194d253268b0a1b?lnk=st&q=1991Oct5.054106&rnum=1#2194d253268b0a1b

Wikipedia (2007) *Xfree86. Web document,* http://en.wikipedia.org/wiki/XFree86

APPENDIX A

System	No. of Developers	Age of Project (years)	Team Organization	Type of Organization Leading Project	Control	Open to New Developers	Mature Technology?
Linux	1722	16	Bazaar	none	Benevolent Dictator	Yes	Yes
Apache	800	13	Bazaar	Volunteer Organization	Committee Consensus	Yes	Yes
MySQL	1000 (est.)	12	Bazaar	For-profit Company	Internal to MySQL AB	Yes	Yes
PostgreSQL	49	22	Bazaar	Volunteer Organization	Committee Consensus	Yes	Yes
Alice	10 (est.)	10+	Cathedral	University Research Team	Internal to Stage3	No	No
OpenOffice.org	151	8	Bazaar	For-profit Company	Committee Consensus	Yes	Yes

A comparison of six open-source development projects.