



The attached material is posted on regulation2point0.org with permission.



J O I N T C E N T E R
AEI-BROOKINGS JOINT CENTER FOR REGULATORY STUDIES

A Developers Bill of Rights: What Open Source Developers Want in a Software License

Alan MacCormack*

**Related Publication 07-12
May 2007**

* This work draws in part from a study funded by the Microsoft Corporation. I am grateful to Lester Chen and David Hering who assisted with data collection and analysis and provided many valuable suggestions for the discussion. Working papers are in draft form. This working paper is distributed for purposes of comment and discussion only. It may not be reproduced without permission of the copyright holder. Copies of working papers are available from the author. Copyright © 2007 by Alan MacCormack.

Executive Summary

In this paper, we study open source developers' perspectives on the nature and structure of software licenses as well as the processes through which these licenses are designed. Recent history has shown that software licensing approaches are critical to the dynamics of the software industry and the open source ecosystem, and thus of interest to the many policy makers and practitioners that follow this part of the global economy. The study is timely, since it informs the debate on the revision of the GPL license, one of the most popular licenses in use. This revision has the potential to shape the software industry for many years to come; hence it is important that the governance process for this revision reflect the needs of the broader software community.

Our study employed structured interviews to capture data on open source developers' opinions about software licenses. We focused on how license choices impact the relationship that exists between open source and proprietary software. Our findings reveal that developers are primarily interested in flexibility and choice when considering a licensing approach. Most developers we interviewed used open source licenses to tap into the open source *development* approach. They chose this option for flexibility in developing a great product, without necessarily espousing any particular philosophy about how the software should be distributed. Developers also generally valued flexibility in the choice of business model for distributing software. The actions of the Free Software Foundation, which is revising the GPL, appear not to reflect the opinions of the broader community, but the agenda of a small minority that may represent as little as 10% of the open source developer community.

Sharing data on the needs and perceived rights of developers, both open source and proprietary, will help the software community, industry experts and policymakers to champion a more flexible and responsive approach to sharing and developing software. Policy makers should work to preserve what has made the software ecosystem successful: innovation, community input and involvement, and developer freedom of choice.

A Developers Bill of Rights: What Open Source Developers Want in a Software License

Alan MacCormack

Overview

Coverage of the debate on the new version of the GNU Public License (GPLv3) has focused on the differing opinions among three groups: Project leaders like Linus Torvalds and other top Linux kernel developers; Foundations like the Free Software Foundation (FSF) led by Richard Stallman; and Large Technology Companies such as Sun, HP, IBM, and Novell. While these three groups are certainly all affected by revisions to the GPL, open source developers are also affected, but have been significantly under-represented in the discussion. In this paper, our objective was to give developers a voice and bring their opinions into the debate. What does this fourth constituency think about open source licenses, the upcoming release of the GPLv3, and the philosophies surrounding open source software? To answer this question, our research explored developers' opinions through interviews. The interviews targeted influential developers who are working on or had worked on a variety of open source projects including JBoss, Apache, Linux Kernel and related tools, MySQL, Apache Geronimo, Snort, Zmanada, XenSource, PostgreSQL and others.

At the center of the license discussion is the FSF, which drives the drafting process for GPLv3. The FSF's leader, Richard Stallman, describes his motivations as wanting "to encourage free software to spread, replacing proprietary software that forbids cooperation, and thus make our society better¹." He views each developer as responsible for protecting the rights of the downstream users:

"Someone who uses your code in a non-free program is trying to deny freedom to others, and if you let him do it, you're failing to defend their freedom."²

Our findings found developers to be interested in flexibility, choice, and their own freedom ("FCF") and less dogmatic in their views. This desire for "FCF" stands out in our six key findings.

¹ Stallman, Richard. "Copyleft: Pragmatic Idealism," <http://www.fsf.org/licensing/essays/pragmatic.html>

² Stallman, Richard. "Why Copyleft?" <http://gnu.mirror.fr/philosophy/why-copyleft.html>

1. Most interviewees use open source licenses to tap into the open source development approach for their project; their focus is on developing a great product rather than a moral imperative to ensure that all software is “free”
2. Most interviewees value the ability to build on the works of others, and believe license incompatibility makes it harder to incorporate other people’s code into their own
3. Developers want the flexibility to vary the license they use for their own code based on need (e.g. so it can be incorporated into other open source or non-open source products); they often choose licenses to increase adoption without concern over ensuring the code is never used for commercial gain or proprietary purposes
4. Many interviewees have worked on both open source and non-open source software, and value interaction between the two
5. Developers often exercise this flexibility to solve practical problems for customers
6. The majority of developers do not support any organization imposing their views upon other developers or abridging other developers’ rights. Most developers are more aligned with the Open Source Initiative’s open source definition, which focuses on allowing users to extend open source creations, but avoids mandating users strictly adhere to the philosophies of upstream developers

Tying to previous research that clustered the open source developer community allowed us to hypothesize how this broader community would feel about the key themes that surfaced and check that the developers we interviewed were a good sample of the broader community. This research found 19% of the community falls into a cluster that believes software should be free.³ Only half of this group espoused opinions opposite to our six key findings. Thus our results suggest the actions of the FSF may only be favored by approximately 10% of the broader community.

Each open source project represents the aggregate work of all developers who have contributed to it. No individual or group of individuals can prevent the desires of the broader community to take a project in a given direction. The process of revising the GPL license represents a paradox to the open source development method as it has been driven by a relatively small number of people who have a disproportionate impact on the developer community, but

³ Lakhani, K.R., and Wolf, R.G. “Why hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects,” in *Perspectives on Free and Open Source Software*, J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (eds.), MIT Press, Cambridge, MA, 2005, pp. 3-21

potentially limited alignment with the community members' goals and objectives. This raises two questions:

1. Does the community need a formal committee like the Linux Foundation to address license revisions?
2. Why isn't the governance system that is used for open source project development used for license revisions?

Introduction

The four main constituencies of the open source community are 1) Development Project Leaders 2) Foundations like the FSF, 3) Large technology companies like Sun, HP, IBM and Novell, and 4) Developers. Significant coverage of the upcoming release of the revised GNU Public License (GPL) has centered on the strong public opinions of the first three constituencies, most notably Richard Stallman, head of the Free Software Foundation (FSF) and Linus Torvalds, the creator for which Linux was named. Large technology companies that use and interact with open source software have also entered the public debate. By contrast, developers, who have no less of a stake, have been under represented. Our objectives were to give developers a stronger voice in the debate on the relationship that should exist between open source and proprietary software and to provide information back to the community to assist them in examining future licenses and revisions.

Given the complexity of this topic, we wanted to gather the opinions of “informed consumers” of open source software licenses – developers who had thought about license choice instead of simply utilizing a “default” option and had considered how licenses affect not just themselves but the broader community. To find these “informed consumers” we targeted developers who had made significant contributions to or were responsible for specific modules of the most widely adopted open source projects. Compared to an occasional contributor, this segment of developers would be more impacted and thus have more incentive to learn about software license issues. Furthermore, as module owners who interacted with many contributors, they were likely to be information hubs who could see the weight of aggregate opinions.

When we interviewed these module owners, we also assessed their motivations for contributing to open source software so we could tie our findings to previous research that

clustered developers based on their motivation for contributing to open source. We did this for two reasons. First, we could use the clustering to check to see if our developer sample was a reasonable match for the broader community. Second, we wanted to test for congruence between the key themes that surfaced and the motivational clusters in order to hypothesize how segments of the broader community would feel about these key themes.

This paper is organized as follows. First, we briefly review key open source licenses, constituents and timelines. We then describe the approach for targeting and selecting interview candidates. Next we report on key findings from the research. Last, we discuss implications for the broader community.

Background

In our interviews, developers often used comparisons to illustrate their opinions. When discussing philosophical differences, they contrasted the Open Source Initiative (OSI) and the Free Software Foundation (FSF). When discussing license terms, they compared a variety of open source licenses. We felt properly understanding their comments required the context of the history and mission statements of the OSI and the FSF and a description of the differences between open source licenses. We also include a history of the GNU Public License (GPL) since the debate about its revision was one catalyst for this research.

Histories

Free Software Foundation

The Free Software Foundation (FSF) was founded in 1985 by Richard M. Stallman who provides the following free software definition: “Free software is a matter of liberty, not price. To understand the concept, you should think of free as in free speech, not as in free beer. Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).

- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this⁴.”

The FSF supports free software development and use, particularly of the GNU operating system, a Unix-like operating system of which the most well-known variant is Linux. The FSF was also responsible for the most widely used open-source and free software license, the General Public License (GPL). Three other main projects of the FSF included:

- The GPL Compliance Lab –investigates potential GPL violations
- The Free Software Directory –lists over 4,000 free software programs
- Savannah –provides development services at no cost to free software developers

GNU Public License

In 1989, Richard Stallman and the FSF released the first version of the GNU Public License (GPL). However, in 1991, two years after the initial release, Stallman took the advice of legal council and the developer community and revised the license, creating version two (GPLv2). The current version of the GPL (the common name for the GPLv2)⁵, has remained unchanged for 16 years. The FSF initiated the revision of the license as they believed that many provisions of the GPL could benefit from modification to fit today's more diverse and complex needs and to reflect lessons learned from the use of version two.

Open Source Initiative

The Open Source Initiative (OSI) was founded by Eric Raymond and Bruce Perens in 1998. This group believed that the philosophical motivations of the Free Software Foundation were confusing and created an anti-business message. By contrast the OSI sought to actively woo the corporate world, in an attempt to “teach business about the superiority of an open development process.”⁶ The OSI is a not-for-profit organization that promotes the benefits of open-source and facilitates cooperation between different members of the open-source community. One of its primary activities is maintaining the Open-source Definition (OSD), which outlines the

⁴ Stallman, Richard. “The Free Software Definition,” <http://www.gnu.org/philosophy/free-sw.html>

⁵ In this paper, we will follow the common usage of referring to the “GPLv2” and “GPL” interchangeably. However, we will explicitly note when we are referring to the third version, GPLv3

⁶ MacCormack, Alan. “Red Hat and the Linux Revolution,” *Harvard Business School Case*, 3/21/2002

distribution terms of open-source. The OSI also approves all open-source licenses and grants an OSI Certified Open-Source Software certification mark to those licenses that uphold the OSD.

Open Source Licenses

The open source licenses most commonly referenced in our interviews were BSD, Apache, GPL, and LGPL. The developers often referred to them to point out what they considered more or less restrictive in a license. The BSD and Apache licenses were considered less restrictive because they did not require derivative works to use the original license, were considered simpler and less complex, and had fewer clauses or restrictions. The GPL and its variant, LGPL, were often used as an example of more restrictive licenses because they were “opposite” to the less restrictive licenses. Table 1 outlines the key characteristics of these licenses.

Table 1 – Open Source License Characteristics

License Group	Characteristics
BSD, Apache	<ul style="list-style-type: none">• Allows code to be used in proprietary software• Does not require that open source versions of the code be distributed• Derivative works may go “closed [source]” or be licensed under a different license
GPL, LGPL	<ul style="list-style-type: none">• Impose substantial requirements on those who create and distribute derivative works, which must be licensed under the same license

Adapted from: St. Laurent, Andrew M., Understanding Open Source and Free Software Licensing, Sebastopol: O’Reilly, 2004

Methodology

Targeted Developers

We targeted developers for our research based on two criteria: the projects to which they had contributed, and their role on those projects⁷. The most well known and broadly adopted open source projects are in the LAMP and JLAMP software stacks, which are commonly used to

⁷ Appendix: Table B shows the exact project mix of the developers we interviewed

run servers. JLAMP consists of an application server (JBoss), an operating system (Linux), a web server (Apache), a database (MySQL), and a scripting language (PHP). The LAMP/JLAMP stacks are often used in the data center, so we targeted developers who had contributed to the various layers including: operating systems, databases, web servers, application servers, scripting languages, virtualization, content management, and security applications. For each application, we used web searches to identify the most relevant projects and our final list included: JBoss, Apache, Linux (Kernel and Tool Chain), MySQL, Apache, PHP, Perl, XenSource, PostgreSQL, Apache Geronimo, Snort, Mondrian, Eclipse, and Zmanda⁸. Given the size of the open source community, we acknowledge this is not an exhaustive list of projects, but felt targeting developers contributing to these projects captures a segment with deep open source experience and informed opinions about the role of licenses.

With regards to the role developers played in an open source project, we targeted developers that were neither the public faces of these projects nor casual contributors. We felt developers in the middle – the “project managers” and “key contributors” – would provide the most helpful insight to the community because the most well known developers (e.g. Linus Torvalds, Andrew Morton) had already published their opinions on licensing and the relationship between open source and non-open source software, and the casual contributor was less likely to have had to think through these issues. This group had also likely interacted with many of the casual contributors and would be able to act as “information hubs.” To find these “project managers” and “key contributors,” we began by identifying contributors for each project from the project web sites, where we found published change logs and acknowledgement or credit pages. We then quantified the contributions of each developer. We counted the number of modules they had worked on and the number of other developers’ modules on which they had signed-off. Additionally, we weighted the importance of the modules based on the prominence they were given on the project’s web site. We ranked developers by quantity and importance of contributions and then recruited those who were in the top half. Finally, during the course of the interviews, we verified their status by confirming they had could check-in and sign off on source code for the project for which we had identified them as a “project manager” or “key contributor.”

⁸ Appendix: Table C shows the open source license used by each of these projects

We sent out 354 emails between February 28th and April 4th. Of the 354 emails, 332 reached their destination and 22 emails bounced. From the 332 emails that reached their destination we received 34 responses for a response rate of 11%. Based on the selection criteria for the developers, and the semi-structured approach we felt that the 34 interviews was more than sufficient to conduct exploratory research to identify the predominant developer opinions on the most critical issues.

Research Methodology

Given the complexity of licensing implications, we felt the topic was not well suited for a structured / quantitative survey. Instead, we used a semi-structured document to facilitate discussion and conduct exploratory research to identify developers' opinions on open source and proprietary software licensing issues. To ensure consistency, we used a common discussion guide for all interviews. Each interview lasted between 45 and 60 minutes. To ensure the responses were a true measure of developers' opinions, we did not use a monetary participation incentive. We conducted the interviews over the telephone between February 28, 2007 and April 4, 2007.

Cluster Identification

Previous research grouped open source developers into clusters based on their motivations. Lakhani utilized a structured quantitative study of 684 developers from 287 distinct projects. He placed developers into four clusters to "provide the best balance of cluster size, motivational aggregation, stability and consistency⁹."

⁹ Lakhani, K.R., and Wolf, R.G. "Why hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects," in *Perspectives on Free and Open Source Software*, J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (eds.), MIT Press, Cambridge, MA, 2005, pp. 3-21

Motivations	Cluster 1 (%)	Cluster 2 (%)	Cluster 3 (%)	Cluster 4 (%)
Work need	91	8	12	28
Non-work need	11	100	0	2
Intellectually stimulating	41	45	69	12
Improves skill	20	43	72	19
Work w/ team	17	16	28	19
Code should be open	12	22	42	64
Beat proprietary software	11	8	9	19
Community rep	14	8	11	13
Professional status	25	6	22	18
Obligation from use	23	20	6	83
Paid for Contribution	86	18	26	32
Total Percentage of sample in cluster	25	27	29	19

Source: Lakhani, K.R., and Wolf, R.G. “Why hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects,” in *Perspectives on Free and Open Source Software*, J. Feller, B. Fitzgerald, S. Hissam, and K. Lakhani (eds.), MIT Press, Cambridge, MA, 2005, pp. 3-21

In the first cluster, developers were most commonly motivated to contribute to open source because of a work need or because they were paid to contribute. All developers in the second cluster were motivated by non-work needs. Developers in the third cluster were most commonly motivated by intellectual stimulation or a desire to improve their skills. Finally, developers in the fourth cluster were most commonly motivated by a belief that they were obligated to give back in return for having used open source code or a belief that code should be free. In this research, we combined clusters two and three because they have the same set of top motivations that are distinct from clusters one and four, and they both have intellectual stimulation as the second highest motivation. We labeled the first group “pragmatists” because of their most common motivation to meet a work need. We labeled the second group (i.e. clusters two and three) “intellectuals” because of the importance of intellectual stimulation as a motivation. Finally, we labeled the third group “philosophers” because their main motivations

were belief-based: a belief that that code should be free and a belief that they have an obligation to contribute due to prior use of open source code.

To identify each interviewed developer's motivations, we first asked them an open-ended, unaided question about the reasons they contributed to open source and to rank those reasons. If they had trouble answering, we then provided them with a list of common motivations from Lakhani's research and asked them to pick the most important one(s). We randomly rotated the order we listed the motivations to remove any bias.

We assigned developers to one of three groups based on their response:

- Group 1 ("pragmatists") – primary motivation was work need or payment for contribution
- Group 2 ("intellectuals") – primary motivation was intellectual stimulation or skill improvement in a non-work context
- Group 3 ("philosophers") – primary motivation was beliefs about code being free or their obligation to contribute based on past use of open source code¹⁰

Analytical Method

In our semi-structured approach, we gathered data on developers' opinions by using a discussion guide. The discussion guide contained predominantly open-ended questions to facilitate a rich discussion. A common discussion guide was used across interviews for consistency.

From the responses, we used an inductive approach to synthesize the developers' responses into key themes. After defining these themes, we looked across responses to identify indicative phrases and responses of a pro or con position on each theme¹¹. We then compared each developer's statements against these indicators to classify each developer as either pro or con on that theme. If a developer provided statements that were mixed (i.e. matched both pro and con indicators for a theme), we examined their responses to related questions. We used the broader context to assign them as pro or con on the theme. For each theme, we found less than three developers who provided mixed responses. The relatively small number of mixed

¹⁰ Appendix: Table B shows the exact breakdown of the assigned groups from our sample

¹¹ Appendix: Table A shows the indicators used to assign developers to positions on each theme

responses gave us confidence our indicators were cleanly telling us whether developers were pro or con on each theme.

We asked both aided and unaided questions to ascertain developers' primary motivations for working on open source projects. Based on their primary motivations, we assigned them to a given Group as described in the Cluster Identification section. For each theme, we cataloged the opinions of developers by Group to identify any congruence between Motivation Group and positions on a theme and to understand what proportion of the broader community would likely hold similar positions.

Our sample size and semi-structured approach were best suited for exploring and unearthing themes and issues. A subsequent study using a quantitative, structured approach would be needed for statistical analyses.

Findings

Through the semi-structured, inductive approach we surfaced a group of key findings about developers' beliefs of licenses and open source software:

1. Most interviewees use open source licenses to tap into the open source *development approach* for their project; their focus is on developing a great product rather than a moral imperative to ensure that all software is "free"
2. Most interviewees value the ability to build on the works of others, and believe license incompatibility makes it harder to incorporate other people's code into their own
3. Developers want the flexibility to vary the license they use for their own code based on need; they often choose licenses to increase adoption without concern over ensuring the code is never used for commercial gain or proprietary purposes
4. Many interviewees have worked on both open source and non-open source software, and value interaction between the two
5. Developers often exercise this flexibility to solve practical problems for customers
6. The majority of developers do not support any organization imposing their views upon other developers or abridging other developers' rights. Most developers are more aligned with the Open Source Initiative's open source definition, which focuses on allowing users to extend

open source creations, but avoids mandating users strictly adhere to the philosophies of upstream developers

Finding One – Developers Value Open Source as a Development Model

Not surprisingly, the developers we interviewed universally expressed their appreciation for the open source development model which relies on a large number of “individual volunteers” to contribute source code, patches, bug fixes and more to open source projects. They noted how the large number of contributors leads to “thousands of eyes” that see each line of code. They applied the label of “faster, cheaper, better” to describe the resulting software.

From our findings, all of Groups One and Two and half of Group Three believed the open source model was only one method for developing software, and other models also had their place. When presented with the FSF’s head Richard Stallman’s perspective that developers should “encourage free software to spread, replacing proprietary software that forbids cooperation, and thus make our society better¹²,” developers from Groups One and Two presented alternative rationale for utilizing the open source software approach. Their rationale centered around four areas: facilitating the creation of technically superior products, realizing the benefits of community involvement, gaining access to wider distribution channels for broader adoption and increasing innovation. The other half of Group Three believed all software should be “free”, open source was always superior, and there was no reason to ever use a closed source model. This minority group stated a moral or philosophical reason for their desire that all code remain free.

In summary, all of Groups One and Two stated their primary motivation for contributing to open source development to be developing a better product that achieves wide adoption rather than philosophical arguments. To them, “The development model matters more than the license”, and “Open source software ensures I don’t have to maintain all the code I write. The broader community can provide support which is invaluable.” These developers also saw the need for other non-open source methods, “Often there are areas where not enough people are interested in it so proprietary software is needed to fill the need.”

¹² Stallman, Richard. “Copyleft: Pragmatic Idealism,” <http://www.fsf.org/licensing/essays/pragmatic.html>

Finding Two – Developers Value Building on Others’ Work

One of the key benefits described by nearly all interviewees for open source software development was the ability to reuse code, methods, ideas, and design principles. In fact, many developers from all three groups mentioned that creating code from scratch would have made it impossible for them to reach the scale of their projects because of resource constraints. Certain types of code were also listed as more critical for re-use. “Embedding occurs most in infrastructure code which is technically challenging to build yourself.” Duplicating components was seen as taking away resources to not only create but also, and perhaps more significantly, maintain the duplicate code - “Time isn’t in writing code, it’s in testing and maintaining it.”

Outside of a few developers who stated, “re-using code is usually more efficient, but not always” the rest of the developers not only saw code re-use as important but also cited the incompatibility of open source licenses as inhibiting code re-use. The most common incompatibility cited was between GPL and non-GPL licenses. Groups One and Two predominately expressed their perception of GPL’s incompatibility through statements like, “Licenses like the GPL are not easily compatible,” “GPL and other open source licenses have trouble working together,” and “GPL is poison because it is totally incompatible and I avoid it like the plague.” These developers described the heart of the incompatibility to be the GPL’s viral nature, which forces any code distributed along side GPL code to become governed by the GPL license. They described this incompatibility as creating a variety of inefficiencies that detract from the stated benefits of the open source method. “At least twice I have taken code from incompatible licenses. We maintained them separately and kept them at arms length,” “When used in commercial products, I’ve had to create work-arounds for functionality to avoid license restrictions,” and “We maintain two code bases. We don’t put GPL code into our house code¹³ base to avoid future restrictions,” all show the impact of license incompatibilities. This incompatibility in effect leads to decreased interoperability through duplication of software development because certain components have to be rewritten for distribution under compatible licenses.

¹³ House code is a term to describe the code a company keeps in an internal repository and is not submitted to the community

Finding Three – Developers want Choice in Licensing

All but one developer agreed that an open source project's license impacts the project's objectives. Developers stated that "License choice is very important," "License is crucial," and "License choice is a huge, huge factor." All but one developer from Groups One and Two, the "pragmatists" and "intellectuals", believed that flexibility in choosing licenses to match objectives was fundamental to a project's success. Developers whose goals included broad scale adoption or use in other projects believed less restrictive licenses like BSD and Apache were critical. "BSD or Apache are best because they provide the freedom to do what you want with the code." Using these types of licenses allowed downstream developers flexibility in how they use the code, even including the option to incorporate the code into a proprietary closed-source product. When asked if they feared someone hijacking their code, developers in these groups responded, "If someone wants to alter my code and use it that is fine as long as you give me credit, I'm not concerned about someone 'hijacking' my code," "I don't care about downstream use of my code, if they use my stuff fine... if they don't contribute good luck w/ bug fixes," and "I'm not worried about someone taking my code."

Groups One and Two did not worry about someone taking their code, and even encouraged others to use it in any way they saw fit. These developers believed licenses got in the way of adoption and often cited the market, not licenses as a reason code stayed open. "In practice it is incredibly difficult to close open code... without the developer's support. It is difficult for a company to support the code." "The community provides invaluable resources to support and maintain the code that most companies don't want to take on internally."

Contrarily, exactly half of Group Three, the "philosophers", believed using the GPL license exclusively was best. Stated reasons included, "GPL license protects my downstream interests and ensures my code stays open," and "I like GPL over BSD because it encourages other vendors to play nice and not lock up code and not contribute." The majority of this group preferred a license to ensure downstream protection and did not want to include a alternative licensing options despite acknowledging potentially lower adoption as a consequence.

Finding Four – Developers Like Interactions between Open and Closed Source

The desired flexibility in license choice expressed by Groups One and Two extended into a desire to accommodate closed source software companies into a broader open and non-open

source software ecosystem. All but two developers in these two groups believed co-mingling and improved interfaces between open and closed source software was important. Many of them mentioned the desire for approved standards which would govern both open and closed source software to improve the pace at which developers could create compatible solutions. “I’m a big supporter of open standards to help open and closed source work together. Compatibility to standards is key to improved interoperability which allows for solutions, and people don’t buy software they buy solutions.” They felt companies contributing and supporting the open source community was a good thing as it benefited the entire ecosystem. “Having proprietary solutions doesn’t make a company unfriendly to open source,” “Companies like to own a piece of code and make a business around it, which is fine,” “This option to keep some stuff closed is important to companies and makes sense.” These comments reflect it does not bother these developers that companies in the ecosystem also protected aspects of their intellectual property through proprietary software development. As long as these companies both take and contribute to the community, open source developers felt companies do not need to open up their entire source code. These two groups felt the interaction between open and closed source software increased the adoption of open source projects by not forcing an “either-or” choice. “I don’t think Linux would be where it is today without allowing non-GPL code to run on top of it.” These groups also felt contributions came in multiple forms and weren’t solely defined as source code. Providing technologies, “Companies subsidizing developers and providing technology is great for the community,” and expanding the reach of the software, “I want as many people to use it as possible” were both cited as valuable contributions that didn’t conform to the narrow definition of contribution.

Group Three, the “philosophers,” were split on this issue. Half strongly felt the need for a “brighter line” between open and closed software. They felt companies playing in both open and closed source software were living up neither to the philosophies of the open source community nor the fundamental requirement to publish all of their source code. These developers shared Stallman’s views and wanted a clear line drawn that forced companies to either fully embrace the free software ideals and make all of their source code available or leave the community. The other half of Group Three acknowledged that while ideally companies would adhere to their philosophies, practically it made sense to accommodate them. This half believed all contributions were good and felt forcing companies to decide would result in some

leaving the ecosystem. To them this was an unacceptable cost because they valued not only the direct contributions of code but also indirect contributions of extending open-source software's functionality or reach.

Finding Five – Developers want Flexibility

The open source developers we interviewed valued flexibility. As described in the previous three sections, they wanted the option to reuse code and ideas from a variety of sources, including closed source software, the option to co-mingle open and closed source software, and the option to incorporate their open source code into closed source software. These desired options share a common theme: greater interoperability between open and closed source software. One way to accomplish interoperability is to dual license their software in situations where their software is useful for another data center application. Another example is allowing downstream developers to incorporate and co-distribute the original open-source software without requiring downstream works to adhere to the same license. This case is illustrated by Apache, which gained greater adoption and reach by being incorporated into Websphere.

The same options developers want to exercise for their work, also solve practical problems for enterprise customers. Research firm Gartner predicts that by 2010, Global 2000 IT organizations will see open source as a viable option for 80 percent of their infrastructure software investments.¹⁴ As this shift occurs, the need for open and closed software to interoperate effectively is essential. Through corporate development projects and the implementation of solutions that combine open and closed source software, customers share the same desires for software as developers: flexibility in licensing to reuse code and ideas from a variety of sources, the option to co-mingle open and closed source software, and the option to incorporate open source code into corporate development projects.

The developers we interviewed recognized the value of interoperability between open and closed source software to customers. When asked whether it is important to accommodate enterprise customers' need to utilize both open and closed source software, one developer replied, "Absolutely" and another stated, "I don't want to cut-off people or commercial entities because of licenses"

¹⁴ Kock, Christopher. "Free Code For Sale: The New Business of Open Source," *CIO*, April 5, 2006

Not only do developers value flexibility to perform actions that are aligned with customers' needs, but they also recognize the options they want are important for solving practical customer issues.

Theme Six – Developers want Choice, not Mandates

Aspects of freedoms, restrictions, and imposition of will through licenses generated strong responses from developers. With the exception of two people, all of Groups One and Two and half of Group Three voiced distaste with anyone imposing their views and abridging other developers' rights. Most developers were more aligned with the Open Source Initiative's open source definition, which focuses on allowing users to extend open source creations, and avoids mandating users strictly adhere to the philosophies of upstream developers¹⁵. This group of developers strongly articulated the need for choice and the need to "let the market decide." While many developers cited displeasure with the patent element of the Novell-Microsoft deal, the use of Digital Rights Management (DRM) to restrict the use of modified open source software, or the enforcement of software patents, (all publicly by Stallman as drivers for the revision of the GPL¹⁶) they did not believe it was the place of the GPLv3 or other licenses to prevent such deals or resolve such issues— "Restrictive licenses are not good for the community. I don't want anybody telling me what I can do with my code." They see the GPL as promoting one viewpoint about users' rights at the expense of their own - "GPL is about freedom of code not freedom of choice... developer is forced to make it free." They repeatedly expressed concern regarding whose freedoms were most important, users or developers, and whether "political views" were entering the license revision process. The GPLv3 was seen as extending restrictions on how people used software code to promote the agenda of the FSF – "I don't want to take freedoms from my customers... new clauses in GPLv3 remove freedoms of how you can use the software. I don't agree with that." "Software licenses shouldn't put restrictions on hardware vendors."

Two people from Groups One and Two and half of Group Three felt free software development was a moral obligation and supported the upcoming GPL revision. Half of Group Three was aligned with Stallman's belief that both Tivoisation and the patent elements of the

¹⁵ See Exhibit 1 in Appendix for full Open Source Initiative Definition

¹⁶ Stallman, Richard. *Transcripts from fifth international GPLv3 conference, Tokyo, Japan*, November 21, 2006

Microsoft / Novell deal endanger the four freedoms¹⁷. Two members of Groups One and Two did not feel aligned with the FSF's philosophies, but did feel the GPLv3 clauses regarding patents and DRM were necessary and beneficial.

Summary

The developers we interviewed clearly articulated their desire for “flexibility,” “choice,” and “freedom” for developers and their code. This is different from user freedoms, which are the freedoms the FSF seeks to protect. Developers see open source as an effective way to develop software, because it gives them access to code or concepts, accelerating development. In general, developers believe license incompatibilities “get in the way” of this key benefit by creating an encumbrance to innovation through their incompatibility. Developers cited a need for license flexibility to achieve project objectives and increase project adoption. Developers believed accommodating closed source code and improving the interoperability between open and closed source code benefited both customers and the broader IT ecosystem. Developers did not want others to force them to use code in a specific way, nor did they want political beliefs to enter their licenses.

Comparing each group's majority viewpoints on these issues, we found Group One, the “pragmatists”, and Group Two, the “intellectuals”, shared very similar beliefs in a desire for flexibility and freedom from a license dictating their actions. They saw the FSF's actions on the GPLv3 directly and indirectly impacting their flexibility and freedom. They believed GPLv3 reduced developers' freedoms and forced a belief system on developers by reducing interoperability and drawing a “brighter line” between open and closed source software. Only half of Group Three, the “philosophers”, disagreed with Groups One and Two. Based on Lakhani's research, Group Three represents 19% of the community. Thus our results suggest the actions of the FSF may only be favored by approximately 10% of the broader community and leads us to ask, should a committee be created with a charter to create and revise open source

¹⁷ Stallman, Richard. *Transcripts from fifth international GPLv3 conference, Tokyo, Japan*, November 21, 2006: Tivoisation: “The requirement is that users must be able to get whatever is necessary so that they can authorize their modified versions to function in the same machine such that they can succeed in operating on the same data, and talking to the same networks.”

MS and Novell: “We were already concerned... that a distributor might receive a patent license which did not explicitly impose limits on downstream recipients but simply failed to protect them... [GPL v3 will] block such deals.” – FSF President Richard Stallman

licenses using a governance model similar to that of the open source development model? Is it contrary to the spirit of the open source community, which relies on the wisdom and view of the masses, to have the governance of licenses controlled by a few individuals whose views run contrary to the objectives of potentially 90% of the people affected by their actions, especially when the community members are the very creators and developers of the software under discussion?

Appendix

Table A – Developer Position Indicators

Key Finding	Indicators developer supported the viewpoint of the finding	Indicators developer did not support the viewpoint of the finding
Most interviewees use open source licenses to tap into the open source <i>development</i> approach for their project; their focus is on developing a great product rather than a moral imperative to ensure that all software is “free”	<ul style="list-style-type: none"> Described open source as a development or innovation model Did not mention needing protection from proprietary companies Mentioned the benefits of both open and closed source methods 	<ul style="list-style-type: none"> Stated belief that all code should be open or free Described open source as a philosophical or moral choice
Most interviewees value the ability to build on the works of others, and believe license incompatibility makes it harder to incorporate other people’s code into their own.	<ul style="list-style-type: none"> Had, wanted, or would re-use other people’s code and saw value in doing so Talked about license incompatibility as a barrier to incorporating other people’s code 	<ul style="list-style-type: none"> Had not re-used other people’s code or saw little value in doing so Was unconcerned about impact of licensing incompatibility on code re-use
Developers want the flexibility to vary the license they use for their own code based on need; they often choose licenses to increase adoption without concern over ensuring the code is never used for commercial gain or proprietary purposes. (e.g. to increase adoption)	<ul style="list-style-type: none"> Chose license to further a project goal Had adopted a dual licensing scheme for their project Chose licenses that allowed maximum flexibility 	<ul style="list-style-type: none"> Chose more restrictive licenses to ensure their code was never used for commercial gain, or chose more restrictive licenses for philosophical reasons
Many interviewees have worked on both open source and non-open source software, and value interaction between the two	<ul style="list-style-type: none"> Had made an effort or thought it was important for their code to work well with non-open source software Valued non-open source companies’ contributions to open source software 	<ul style="list-style-type: none"> Wanted open source software to stand on its own Supported a divide between open source and non-open source software
Developers often exercise this flexibility to solve practical problems for customers.	<ul style="list-style-type: none"> Made development or licensing decisions that increased interoperability 	<ul style="list-style-type: none"> Made no effort to increase interoperability between open source and non-open source software
The majority of developers do not support any organization imposing their views upon other developers or abridging other developers’ rights. Most developers are more aligned with the Open Source Initiative’s open source definition, which focuses on allowing users to extend open source creations, but avoids mandating users strictly adhere to the philosophies of upstream developers.	<ul style="list-style-type: none"> Felt the FSF’s philosophy was not aligned with their own Felt the FSF’s actions was looking out for the FSF’s interests, not developers Supported flexibility for developers 	<ul style="list-style-type: none"> Felt the FSF’s philosophy was aligned with their own Felt the FSF’s actions helped developers Supported mandates to protect users

Table B – Developer Demographics

Project	Developers Interviewed	Group	Developers Interviewed
Amanda	2	One – “Pragmatists”	19
Apache	4		
Apache Geronimo	3	Two – “Intellectuals”	8
Eclipse	1	Three – “Philosophers”	7
GCC	4		
Toolchain			
Jboss	3		
Linux	7		
Kernel			
MySQL	1		
Perl	2		
PHP	2		
PostgreSQL	2		
L			
Snort	2		
XenSource	1		

Table C–Projects by License

Project	License
Linux, MySQL, ZenSource, Snort, Zmanda	GPLv2
JBoss, Zmanda	LGPL
Apache, PHP*, Apache Geronimo, Zmanda	Apache
Perl	Artistic
PostgreSQL	BSD
Eclipse	Eclipse
MySQL	GPLv2 + Commercial

* Self-described as an “Apache-style license”

Source: <http://jboss.com/opensource/lgpl/faq>; www.linux.org;
<http://www.apache.org/licenses/>;
<http://www.mysql.com/company/legal/licensing/>; <http://www.php.net/license/>;
<http://www.perl.com/pub/a/language/misc/Artistic.html>;
<http://www.xensource.com/company/legal.html#d>;
<http://www.postgresql.org/about/licence>; <http://www.apache.org/licenses/>;
http://www.snort.org/about_snort/licenses/;
<http://sourceforge.net/projects/mondrian>;
<http://www.eclipse.org/org/documents/epl-v10.php>;
<http://www.zmanda.com/amanda-license.html>

EXHIBIT 1: The Open Source Definition

Source: <http://www.opensource.org/docs/osd>

Introduction

Open source doesn't just mean access to the source code. The distribution terms of open-source software must comply with the following criteria:

1. Free Redistribution

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. Source Code

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed. Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. Derived Works

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. Integrity of The Author's Source Code

The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. No Discrimination Against Persons or Groups

The license must not discriminate against any person or group of persons.

6. No Discrimination Against Fields of Endeavor

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. Distribution of License

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. License Must Not Be Specific to a Product

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed should have the same rights as those that are granted in conjunction with the original software distribution.

9. License Must Not Restrict Other Software

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

***10. License Must Be Technology-Neutral**

No provision of the license may be predicated on any individual technology or style of interface.

EXHIBIT 2: Aggregate Developer Quotes

Theme One - Open Source Development Model

Pro	Con
The development model matters more than a license	I don't like proprietary companies taking code and not letting community see it
Open source is simply a practical means for developing software	Always more smart people outside a company than in
Open Source fundamentally promotes innovation	Ideally all software would be open source
Ok for MS, Novell or others to take code	I agree w/ philosophical point of view about code being open. Mac OS and Windows gave me heartache
Makes economic sense to use open source, not about morals	
Easy to use and try	
Fast adoption and testing	
Leverage component reuse Some software should stay closed	
Open source development can make for a better product because people don't work on stuff they don't care about.	
I'm more effective working on open source software, but I've also contributed to proprietary companies	
I'm not a zealot or a purist	
I support BSD style licenses for networking applications and allowing companies to keep certain items proprietary	
Impossible to do at this scale w/o open source	
Open source doesn't fit every software model, lots of areas where this model	

doesn't work because can't get money back	
Open source is not just having source code it is having community involvement	
I'm very pragmatic Open source has its strengths but other options exist	
Effective way to communicate w/ other people, there are downsides to OS	

Theme Two - Innovate on others' work

Pro	Con
I want access to proprietary drivers and believe it would help projects like Linux	Re-using code is usually more efficient but not always
I see GPL as stopping me from using other code	People are told not to read patents, it is better to say you didn't know
It is a drag to re-engineer because of license compatibility	The projects I have worked on haven't had to access protected work much
I would like access to IP or other innovations	Avoid looking at anything not under open source licenses
At least twice [I] have taken code from incompatible licenses. We maintained them separately and kept them at arms length	Ideally you are calling someone else's code not embedding it, GPL and other OS licenses have trouble working together
Code re-use is important benefit to Open Source community	License challenges could be a good thing as they enable good things to happen
Java code that is GPL'd is challenging...it is cumbersome and requires work-arounds	I don't think it is important to take a piece of code from another project. Copying code slows you down because it's never perfect
Code reuse leads to faster time to market	
Plug-able libraries are important	
All the time have to re-engineer	
Licenses get in the way of component re-use	
Re-usability of code is important	

If functionality already exists, I'll use it. If I need to borrow, then I'll call vs. copy. If I don't have access, then I'll copy it. If I can't find it, then I'll write. Someone else's code probably has been bug tested. If I write it, then there's more code and the software gets bloated.	
License incompatibility reduces code re-use to some extent	
Things that make good sense for software are avoided for legal issues	
More liberal licenses are easier to deal with	
I want access to proprietary drivers and believe it would help projects like Linux	
Yes, I would pay for access to proprietary technology if it helped me fix my code	
Embedding is very important.. many people are better at specific things, don't want to re-invent if I know it works well.	
...licenses get in the way of modifying existing code base lowering innovation	
I would like access to IP or other innovations	
I see GPL as stopping me from using other code	
GPL code is compatibility issue	
Copyrights limit work more than patent	
License compatibility is limiting	
Where licenses permit, I've pulled in code	
I often have to change other's code for my enabling code to work	

Theme Three - Vary license choice

Pro	Con
We need an open source license that pays attention to the needs of companies	Licenses have some affect on re-usability but not on interoperability... May just require additional code
Choice of a license helps for objectives	As an academic I have become convinced to release code w/ no license to avoid academic propriety issues
I have run many projects and have never chosen a GPL style license as I think they are too limiting	Current version of GPL meets my needs
Apache license is freedom for people to share or not, it is about choice	GPL allows software to live and is also a great license for commercialization of technology
Dual licenses provide flexibility for commercial and non-commercial use	Licenses have some affect on re-usability but not on interoperability... May just require additional code
Dual license gives choice. Customers who want to integrate use the non-gpl'd version	Would always use GPL2
License choice helps a project objective. GPL helped Linux and BSD would have hurt it	One generic license for OS would help standardize the landscape, license choice is critical to adoption
Jboss was smart to use LGPL as it allows for embedding	I would prefer everything to be GPL to keep code open but GPL was not the key to Linux success
BSD / MIT [licenses] are easiest for a developer	
I don't care about downstream use of my code so I choose least restrictive license	
As a programmer I want to write code and have others use it... licenses like Apache are easy	
I have been in the position of asking others to re-license their code so I could use it	

Choice of a license helps for objectives	
Jboss was smart to use LGPL as it allows for embedding	
I want a license that attracts enough developers but also want companies to use the code	
I want as many people to use it as possible	
Each license has its proper place and business model they support	
Lack of restrictions is important to me so I chose BSD	
Everybody has their own needs. Choosing a license that works for you is fine	

Theme Four - Interaction between open source and non-open source

Pro	Con
A more bright line that makes people choose between open and close source isn't good	License barriers can be good because duplicate works creates new solutions
I want access to proprietary drivers and believe it would help projects like Linux / Co-mingling is good and the customer benefits	Community only works if people contribute to it.. .from a pragmatic point of view it is nice to have companies contributing, but if they don't go along w/ the community I wouldn't miss them
We were ok w/ MS and Netscape taking Apache code, we wanted adoption	No, open and closed source code working together is not important to overall success of Open Source
The choice between Proprietary and Open software isn't black and white, both are ok	Companies need to choose between open and closed source. This is a revolutionary change not an evolutionary one
I believe co-mingling is good and licenses impact the degree of co-mingling	Agree w/ FSF ethos
The two worlds (open / closed) co-exist and that is good	I have a general preference not to work with closed source. I doubt access to

	proprietary IP would be worth paying for
There is no reason to exclude ourselves from working w/ closed source software	I would prefer a cleaner distinction between open and closed source, Technologies are open source, but soln's are business side
Open source doesn't fit every software model hence the need for closed source	Mixed source is ok, but companies that haven't embraced OS yet have to make a clear choice
Co-mingling is important for Linux Adoption (Oracle working on Linux is a good thing)	
I think being part of Websphere helps Apache.	
Having [closed source] specs would assist greatly	
I very much believe in the hybrid model	
Open standards are important	
Big supporter of open standards to help open and closed work together	
Definitely support mixed source interaction	

Theme Six - OSI vs. FSF

OSI	FSF
FSF isn't interested in clarifying terms in the GPL so they continue to exert power	Appropriate for FSF to look at the new technology changes going on in the world and revise the GPL
FSF should listen more to other stakeholders	Clarification of patent claims is an important reason to move to GPLv3
I don't want to make those types of deals (MS / NOVELL) difficult, because the future is unknown	The FSF's philosophies completely align with my own

FSF is like early Ford... You can have any car you want as long as it is black e.g. any freedom as long as it is Stallman's freedom / FSF is not the sole moral compass for the community	I personally like a GPL license better as it insures the code grows in the community
My main customer is other developers and their needs are very important to me GPLv3 confirms what people outside of the community think of us.. That we are trying to destroy IP and force software socialism. I don't like that	If you contribute to BSD, your code can be made closed source. You've lost control over it. By contrast, the GPL gives you back as much freedom as you gave.
FSF thinks they represent the work of the whole community and they don't	I see conflict from TIVO and how they aren't abiding by the spirit of the GPL
I don't like the MS - Novell deal but I don't think terminating someone's right to distribute GPL is ok	
Hardware vendors definitely should have the ability to protect it	
I don't think there is a licensing solution to IP issues	
Already concerned about whether GPL2 is good for the community	
Don't want to start a war over IP	
GPL (FSF) are incredible hypocrites because they espouse the needs for freedoms but tell you how to use your own code	
FSF are completely insane	
If my motivation is to get my source code used, I believe I am better off using a less restrictive license like the BSD. If my motivation is some sort of activism ... then I would choose a license like GPL that forces other people to share my vision. ... [but] I don't want to have to subscribe to someone else's vision of utopia. / I like	

their [FSF's] promotion of open source [as a concept] but not their vision of how open source should work	
I don't like the anti-business efforts	
FSF should listen more to other stakeholders	
Don't want a library that contributes 5% of the code to dictate the project's license	
Not happy how my previous GPL projects will be affected. Downstream could change them to GPLv3 w/o my permission	
FSF has done a good job promoting open software but they are too religious	
FSF isn't interested in clarifying terms in the GPL so they continue to exert power	
GPL is too strict for my needs	
GPLv3 generally speaking is becoming more restrictive which I don't like	
Personally I don't want to control how people use my stuff	
I'm not too interested in GPL2 or GPLv3, people in GPL are very religious about free software and I'm not	
Companies shouldn't be forced to open up all their source code	
GPL forces you to open everything which discourages companies w/ IP	
I went to Open Source because I was forced out by proprietary companies, I like having flexibility on what I can choose and not choose to do. I don't want to force someone else to my point of view	