Open Source Licensing across Package Dependencies

Maria Kechagia, Diomidis Spinellis and Stephanos Androutsellis-Theotokis
Department of Management Science and Technology
Athens University of Economics and Business
Athens, Greece GR-104 34
Email: mkehagia@dmst.aueb.gr, {dds, stheotok}@aueb.gr

***Abstract*—Licensing dependencies among open source software (oss) packages reveal a lot about software compatibility relationships and the practicalities of oss licensing. There is, however, limited information on these in the literature. In this paper, we discuss various aspects of oss licensing, and we present an empirical study on FreeBSD ports collections concerning their licensing dependencies, in an attempt to identify specific patterns. Our results highlight different types of dependencies, that could be used to explain, or even guide the license selection process of oss projects.**

***Keywords*-oss; licenses; dependencies; FreeBSD ports**

I. Introduction

Open source software (OSS), can be freely used, modified, and distributed, provided certain restrictions are observed regarding its copyright and the protection of its status as OSS. These rights and restrictions are expressed through the software's license, i.e. a contract between the software owner(s) (the licensors) and its prospective users (the licensees). OSS licenses come in different flavors, but in general they make available the software source code and they permit the creation of derivative works as well as (usually) the non-exclusive commercial exploitation of both the original and the derivatives [1].

The objective of the work we describe in this paper is to gather empirical evidence regarding the effect of OSS licenses on the way the software is actually used. We work on the FreeBSD ports collection; a set of more than 20,000 software packages distributed in source and binary form together with the FreeBSD operating system. Many packages use facilities provided by others, thus forming an intricate web of compile and run–time dependencies. By examining the relationship between these over 70,000 dependencies and the licenses under which the corresponding packages are distributed we can see whether some common usage patterns are associated with specific licenses. Specifically, we aim to answer the following two research questions.

1) Are some license types particularly conductive to reuse?
2) Do the licenses across dependencies follow an order associated with the permissiveness of each license?

Results could explain or even guide the selection of a particular OSS license.

II. Concepts and definitions

Before discussing the different types of OSS licenses, we briefly introduce the background concepts delineating the degrees of freedom available while distributing the product of any intellectual activity, including software.

A. Intellectual property, copyrights and patents

The term intellectual property is used to encompass a wide range of areas of law, including copyrights, patents and even trademarks [2]. These are all means used to encourage private investment in research, technology and innovation, by ensuring that innovators will be able to get private returns for their work.

Since inventions, and software code, can be copied and reproduced, it is argued that patent or copyright protection is necessary. However the free software community expresses considerable concerns about the use of such protection in software, claiming that the software community and society in general benefit less from this restrictive approach, relative to having the knowledge that the innovators have created free and available to all [3].

B. Public domain

At the other end of the spectrum, by labeling software as Public Domain, its owner declares that there is no copyright protection and no distribution or licensing restrictions. Anyone is free to copy, modify, distribute or sell the software, without any permission being required [4], [5].

There is a major misconception equating OSS with public domain [6]. However OSS is not public domain software. It is copyrighted and distributed under a license, just a license that gives the users more rights than they are typically used to.

C. Open source and copyleft

Open source lies in between allowing a software work to fall completely in the public domain (thus relinquishing any notion of ownership), and protecting it under copyright or patent law. All open source software licenses share two characteristics:

In Costas Vassilakis and Nikolaos Tselikas, editors, *PCI 2010: Proceedings of 14th Panhelenic Conference on Informatics*, pages 2732, Los Alamitos, CA, USA, September 2010. IEEE Computer Society. (doi:10.1109/PCI.2010.28).

They waive the right to earn license fees from distributing the software, and they incorporate the condition that the source code will be made available to licensees.

Copyleft (a play on the word copyright) is a form of open source licensing that grants the right to reproduce, adapt or distribute software, however imposes the restriction that any derivative work will be released under the same license. In this way the software and the freedoms applied to it become inseparable [2]. Copyleft licenses are therefore a subset of all OSS licenses. They are further distinguished according to how restrictive they are, and often labeled as strong copyleft, weak copyleft (as opposed to the non-copyleft ones).

There are two main movements that promote OSS and certify licenses as free and/or open source software: The Free Software Foundation (FSF)[1] and the Open Source Initiative (OSI)[2].

III. Licenses

Figure 1 summarizes the main OSS licence categories, their relationship to other software license types, their main features and some representative examples that will be discussed below in more detail.

A. The GPL and copyleft licensing

The GNU General Public License (GPL) was created in the mid 1980s by Richard Stallman for the GNU Project distribution from the Free Software Foundation [2], and its terms provided much of the foundation for free software development. A key feature of this license which contributed to its widespread adoption, is the licenses "viral" nature, as it enforces the source code of any derivative work from a GPL-licensed software to also be released under the GPL.

B. The lesser-GPL and other weak copyleft licenses

1) The LGPL license: The GNU Lesser General Public License (LGPL) also known as the Library GPL, is a derivative of the GPL proposed by the FSF, intended for use mainly with software libraries. Its main differentiator from the GPL is that an LGPL licensed program or library can be incorporated within a proprietary program, or more generally one that is not licensed under the LGPL.

2) The MPL license: In 1998 Netscape proposed a beta version of the Netscape Public License (NPL) for public comment. Based on feedback received they refined it into a second license, the Mozilla Public License (MPL). Similar to the LGPL, the MPL allows the creation of larger, derivative work, including proprietary code that is not required to be published. Still, any changes to the original source code must be made available to the community [12].

[1]http://www.fsf.org/. Note: All web URLs in this paper last accessed in June '10.

[2]http://www.opensource.org/

3) The Artistic license: The Artistic License (AL) was created by Larry Wall in 1991 for Perl, as he felt that the terms of the GPL (under which Perl was released until then) were too restrictive. The goal of the AL was to allow Perl to be used in commercial packages [12]. The AL is very similar to the GPL, but being a weak copyleft license it doesn't require distributing derivative works under the same terms [13]. It basically allows the programmer to do anything they want as long as the changes are published and described in the source code, or all executables are renamed and the differences are documented, thus allowing the original author to maintain artistic control [12].

4) Other weak copyleft licenses: Various other licenses have been proposed that embrace the weak copyleft approach, including the Sun Industry Standards Source License (SISSL)[3], the Sun Public License (SPL)[4], as well as the IBM Common Public License (CPL)[5] and its derivative Eclipse Public License (EPL)[6].

C. The BSD and other non-copyleft licenses

1) The BSD license: The Berkeley Software Distribution (BSD) license was originally used for the release of significant portions of a Unix implementation by the University of California. Since then, a fair amount of open source software was distributed under this license.

The BSD license allows covering derivative works under different terms or licenses, as long as the necessary credit is given to the original work. Being one of the main non-copyleft licenses, it essentially imposes no requirements on developers working with source code released under a BSD license. Any modifications can be made and redistributed in any manner they choose. And, as opposed to weak copyright licenses, there are no incentives or requirements to contribute the modifications back to the community [12]. It does however require that any acknowledgments of previous contributor's work are retained [14]. Finally the BSD license also includes a no-endorsement clause saying that the names of the originators and contributors cannot be used to endorse products derived from the source code [12].

2) The Apache license: The Apache License, a derivative of the BSD and similar to the MIT/X11, was proposed by the Apache Software Foundation (v2.0 written in 2004). It makes clear that the licensing of derivative works under other licenses is permitted so long as the terms of the Apache License v2.0 are complied with (this is implied but not specifically spelled out in the MIT and BSD Licenses).

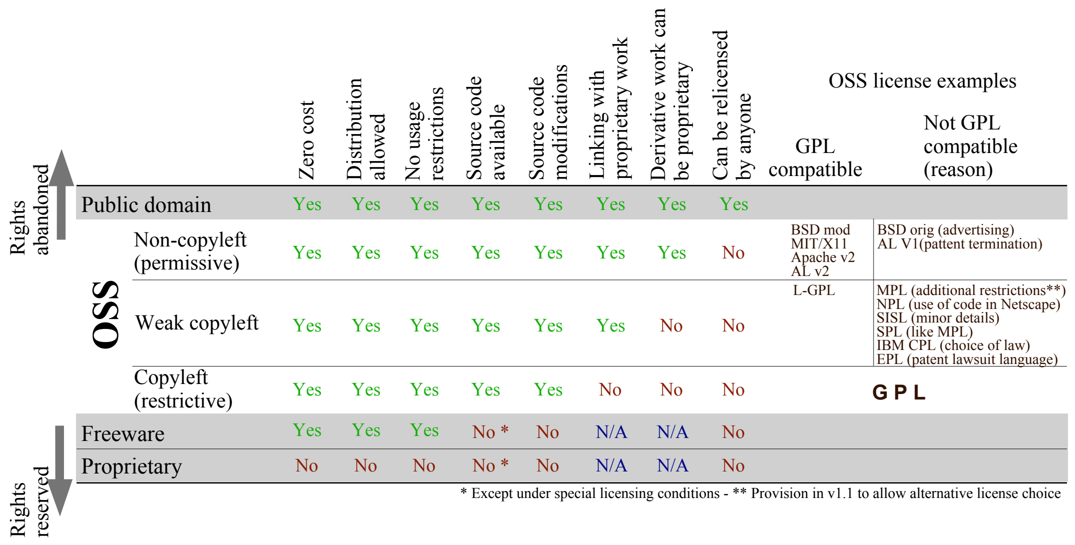
3) The MIT/X11 license: Another non-copyleft license, the MIT/X11 actually predates the BSD (it was written in 1987 for the X Windows System source code). The two licenses are very similar (one difference being that the MIT/X11 does not include the no-endorsement clause).

[3]http://www.opensource.org/licenses/sisslpl.php

[4]http://java.sun.com/spl.html

[5]http://www.ibm.com/developerworks/library/os-cpl.html

[6]http://www.eclipse.org/legal/epl-v10.html



Rights abandoned

		Zero cost	Distribution allowed	No usage restrictions	Source code available	Source code modifications	Linking with proprietary work	Derivative work can be proprietary	Can be relicensed by anyone	OSS license examples: GPL compatible	OSS license examples: Not GPL compatible (reason)
Public domain		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		
OSS	Non-copyleft (permissive)	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	BSD mod MIT/X11 Apache v2 AL v2	BSD orig (advertising) AL V1(pattent termination)
OSS	Weak copyleft	Yes	Yes	Yes	Yes	Yes	Yes	No	No	L-GPL	MPL (additional restrictions**) NPL (use of code in Netscape) SISL (minor details) SPL (like MPL) IBM CPL (choice of law) EPL (patent lawsuit language)
OSS	Copyleft (restrictive)	Yes	Yes	Yes	Yes	Yes	No	No	No	**G P L**	
Freeware		Yes	Yes	Yes	No *	No	N/A	N/A	No		
Proprietary		No	No	No	No *	No	N/A	N/A	No		

Rights reserved

* Except under special licensing conditions - ** Provision in v1.1 to allow alternative license choice

Fig. 1. A categorisation of the OSS licenses with respect to other software license types, their main properties, and some characteristic examples. Based on material from [7], [8], [4], [9], [10], [11].

IV. Factors in selection of appropriate license

Various considerations will affect the decision over what license to apply to an OSS project or program.

First of all, it is generally advised to go with one of the existing and empirically tried licenses, rather than draft a new one [12]. Using a well-known and trusted license will give the users confidence and clarity regarding what uses of the software are allowed. On the contrary, an obscure, overly complicated and rarely used license will probably create confusion and ambiguity. And constructing a license from scratch, requires a lot of experience and knowledge of legal matters, so it is not advised except for special cases.

Furthermore, in various cases the choice of license may be limited by pre-existing software used in the project, and its own licensing scheme. For example if pre-existing BSD-licensed software is used, the project team has the freedom to select any license, provided they respect any requirements regarding notifications, disclaimers etc. But if GPL-licensed software is used, then the only option would be to use the GPL for the resulting project as well.

Provided there is freedom of choice, the most important factor is the permissiveness of the license, i.e. to what extent it allows re-using derivative work to be licensed under other schemes [1].

Other factors to be concerned include specific project characteristics (e.g. topic, natural language, environment and operating system, development stage) [9], as well as developers' motivations and community's needs [10].

- Topic and audience. It is argued that software aimed at developers, system administrators, or more generally technically proficient audiences, as well as projects on topics that target sophisticated peers, are more likely to be licensed under permissive licenses (e.g. BSD, MIT, Apache, etc.). The reasons include the strong community appeal of such software, as well as the fact that developers involved in these projects are often motivated by career enhancement opportunities, and will therefore be favorable to licenses that will allow them to demonstrated their skills to sophisticated audiences.
- Environment and operating system. Projects based on more commercial platforms and operating systems are likely to employ less restrictive licenses.
- Industrial involvement. If companies have a significant involvement in the project, then they are likely to be reluctant to adopt a strong copyleft license.
- Commercialization goals. If the option of including the project's software in some commercial project is considered, then a non-GPL license that will permit this will be required.
- Protection from copying. If a project feels that it needs to protect its code from other groups that may copy it and utilize it in their own products, then a license that would prevent this, or at least require that they return to the community their changes, would be preferable.
- Attitude. The degree of restrictiveness of the license of choice will depend on whether the developers and project communities believe in the right to redistribute one's work under licenses of their choice.
- Motivation. Various studies (including [10] analyze theoretically and empirically how the developers' possible intrinsic and extrinsic motivations (e.g. the problem solving challenge, recognition by peers, monetary incentives and future employment) may affect their choice of license for their software. It is generally found that more permissively licensed projects attract more highly skilled programmers who may want to maintain intellectual rights for their work, or simply for ego gratification, and stimulate more contributions. More restrictively licensed projects, on the other hand, ensure access to everyone's

contributions and are favored by communities with less free-riding [11].

V. Concerns and risks of OSS licenses

Various concerns have been voiced regarding the adoption of different OSS licensing schemes.

A concern regarding the use of licenses that allow the combination of open source and proprietary software is that they effectively undermine the concepts which the OSS movement advocates [15]. Furthermore, it allows a competitor of a non-copyleft licensed project to take the source code and build a proprietary, non open source product (e.g. Apple's Mac OS X and FreeBSD) [11].

Combining OSS released under different licenses also requires attention to compatibility issues [13], [16]. As a general rule, OSS released under different licenses can be combined to yield an outcome under a license at least as restrictive as the two original ones. However there are exceptions to this rule, and the particularities of each license need to be carefully taken into account. For example software under the MPL license cannot be redistributed under licenses that impose restrictions not present in MPL. As a result, MPL software is, in principle, incompatible with GPL. However even in this case, a provision exists in the MPL license that allows a program or parts of it to offer a choice of another license as well[7], thus partially overcoming this restriction.

Unlike proprietary software, projects under OSS licenses are, to different degrees, also unprotected from forking danger [1]. The GPL license reduces the danger of forking by enforcing that the derivative work will remain under the GPL. But under other licenses, and mainly non-copyleft ones (such as BSD and Apache), the community is unprotected from developers forking off and continuing with non-OSS development. To an extent the community relies on the reputation of the developers to avoid forking, as well as on other measures.

Commercial software development firms may feel that there is a risk involved in incorporating OSS code in their products, due to the lack of clarity in some definitions [17], [16], [4]. For this reason, many corporate licenses (e.g. MPL, CPL, EPL, etc.) have been created [14]. As discussed above, some OSS licenses only allow the reuse of software if the derived work is also under the same license (most notably GPL). But the definition of derived work may not be clear enough to dictate how the original OSS could be used. For example it has been suggested by some that if a GPL-licensed program is dynamically linked to a proprietary library (or vice-versa), the result would not need to be licensed under the GPL, as the two programs retain distinct existences. However the FSF does not accept this position, and argued that in such cases the LGPL should be used instead of GPL [4]. Possible strategies around this may include clearly separating, at the architectural level, the pieces of the resulting software product that rely on OSS code from other parts, and license the former as OSS and the latter as proprietary [18], [19].

In any event, the use of legal advise would be recommended [17], or alternatively "packaging companies" could be employed to serve as intermediaries between the OSS community and the proprietary software house and undertake various technical responsibilities, as well as legal, licensing and intellectual-property rights (IPR) issues. [19]. Alternatively, permission may be explicitly requested from the OSS project owners to include parts of the code within a proprietary product [18].

VI. Method and Tools

FreeBSD [20] is a sophisticated operating system available for a number of modern architectures. It is a complete operating system (rather than just a kernel, like Linux) derived from BSD Unix, the version of Unix developed at the University of California, Berkeley. FreeBSD, known for its stability and reliability, runs the servers of large portals like Yahoo and hosting providers like the Host Department; parts of it also form the basis for Apple's Mac OS X. A strength of FreeBSD is its *ports collection*: third party applications ported to install and run under FreeBSD in an easy and seamless manner. Our research is based on an April 2nd 2010 snapshot of the FreeBSD ports. This contains 21,458 ports organized into 64 categories, starting from accessibility applications and ending in X11 window managers.

Where possible, the FreeBSD ports are designed so that one can install them by compiling their (suitably patched) source code. This allowed us to download the ports' archives (48GB) and unpack them into a 173GB fully searchable source code tree. Many ports use libraries, tools, or data provided by other ports (see example in Figure 2). The *Makefile* associated with each FreeBSD port defines these build or run–time dependencies, so that the port management commands can install all the infrastructure required for a port to run. By executing a special *Makefile* target for each port we gathered a set of 71,872 port dependencies. We were then able to use the *GraphViz gvpr* tool [21] to find the transitive closure of all dependencies leading to a given port and thereby obtain a measure of its reuse.

In contrast to the tracking of port dependencies, the license associated with each port is not marked in the FreeBSD ports system in a dependable way. Given the legal ramifications of ascertaining the license of a third party application and thereby inferring detailed restrictions on its use this lack is understandable. We therefore used two methods to find each port's software license. The first method involved finding the most popular OSS licenses, according to the ranking of the sourceforge.net portal. We then went through their text, and located a key phrase that uniquely identified that license. With those signature phrases at hand we could search all the files of a port using *fgrep*, which relies on the extremely efficient Boyer-Moore-Gosper algorithm [22], for instances of that phrase. Our second method involved using the *ohloh*[8] project's *ohcount* tool to determine the license associated with

[7]http://www.gnu.org/licenses/license-list.html#GPLIncompatibleLicenses

[8]www.ohloh.net

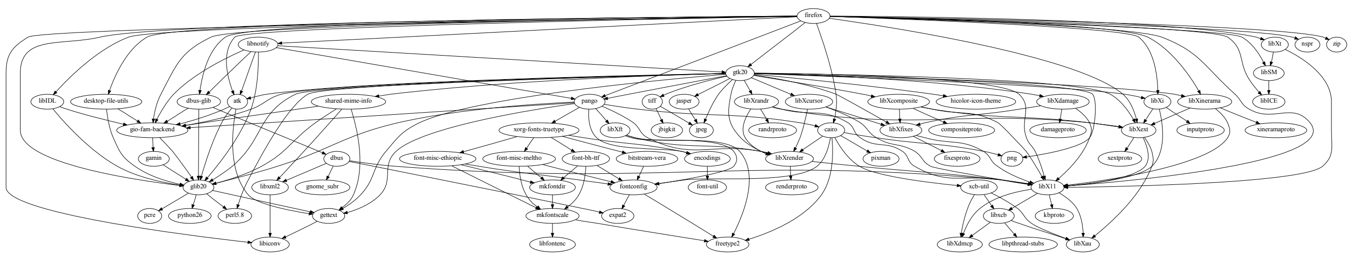


Fig. 2. Run-time dependencies of the Mozilla Firefox web browser.

each port's file. Under both methods we sorted the results by the licenses identified and assummed that the the port's license was the one associated with the largest number of files.

VII. Results

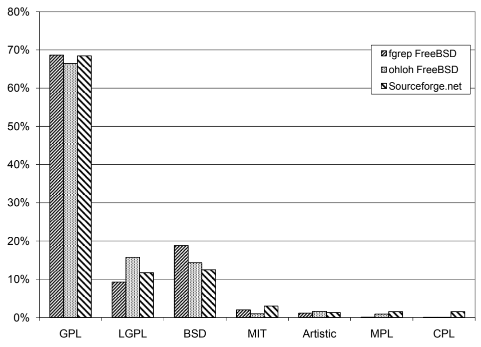


Fig. 3. Distribution of open source licenses.

Out of a total of 21,458 packages our method identified the license of 12,899. We used ohloh's tool to verify our results and found the two tools to agree in 83% of the cases. We kept the results of our method rather than those of the ohloh's tool, because our method could identify the license in a higher proportion of the packages (62%) than that identified by the ohloh tool (56%). The license distribution as identified by our method, ohloh's tool, and the sourceforge.net reports appears to be roughly similar (see Figure 3), which further validates our method.

A. License Types and Reuse

We looked at the average number of a port's total number of afferent transitive dependencies in each set of ports distributed under a given license. The results of the most commonly used licenses appear in Table II. Contrary to our expectations, ports distributed under the GPL license, which imposes more restrictions on the way the software can be used, are used by more ports than those distributed under the LGPL license, which is explicitly used to encourage reuse, or the BSD license, which imposes minimal restrictions.

TABLE I
Average transitive number of afferent dependencies according to a port's license.

License	Dependencies
MIT	249
GPL	108
LGPL	101
(Overall average)	66
BSD / Apache	25
Artistic	7
Apache v2	3

TABLE II
Number of relationships between packages according to their license.

Main port	Reused port	Dependencies
GPL	GPL	18596
GPL	LGPL	10612
GPL	BSD / Apache	2704
LGPL	GPL	2703
LGPL	LGPL	2150
BSD / Apache	GPL	1805
BSD / Apache	BSD / Apache	1166
GPL	MIT	1093
LGPL	BSD / Apache	639
BSD / Apache	LGPL	600
MIT	GPL	482

B. Licenses across Dependencies

The second question we aimed to answer was whether licenses across dependencies follow an order associated with the permissiveness of each license. Here the results fit more what one would expect. GPL and LGPL–licensed projects use other projects based on the permissiveness of their licenses. On the other hand, BSD-licensed projects appear to be in an uneasy relationship with others, with no clear preference to licenses according to their permissiveness.

VIII. Related Work

To our best knowledge, the number of empirical studies concerning licensing dependencies is limited. Two papers are most closely related to our work.

In the first one, German [23] introduced the question of how licenses affect the development of dependency graphs. Dependency graphs are a way to depict relationships among different entities. Thus, they are very useful for studies, like German's and ours, to specify dependencies among a distribution's packages, and study the licensing ones, in depth.

In the second one, German et al. [24] present a method for license identification, based on sentence-matching, similar to ours, and implement a corresponding tool. Such tools are vital for analysing licenses, and a number of projects, like OSLC[9], ASLA[10], and FOSSology[11], are currently providing some of them.

IX. Conclusion

The license types associated with the nodes joined by each edge of a software package dependency graph reveal a lot about the practicalities of OSS licensing. Although at a simple aggregate view a license's popularity dominates the relationships between packages, by pairing licenses together we found that dependencies exploit, follow, or avoid licenses according to their intended use. Specifically, GPL-licensed applications do indeed use LGPL-licensed components, and BSD-licensed packages are more a target than a source of dependencies.

Based on our results, it seems that a license's popularity should be highlighted in the list of factors affecting OSS licensing selection. Developers should pay attention to the fact that the more popular the license of their project is, the more dependencies it can develop to other licenses and projects, no matter how permissive the license is by itself. Additionally, the permissiveness of a license affects the preferences of a license towards others. Taking into account this factor, developers can forecast which and how many licenses and projects their project can be linked with. Thus, our research recommends the examination of both popularity and permissiveness of an OSS license.

Our simple study opens more questions than it answers. A more detailed analysis should factor each license's popularity when looking at the frequency of license pairs across dependencies. Improving the effectiveness and accuracy of our license detection method could allow us to determine the effective license of each application based on the licenses of its components, and also to examine the use of incompatible licenses. A more intriguing possibility worth studying is whether protective licenses, like GPL, and permissive licenses, like BSD, form closed ecosystems where dependencies among protective and permissive licenses are explicitly avoided.

Acknowledgment

The authors would like to thank Georgios Gousios for assisting us with the required computational infrastructure.

References

[1] A. M. S. Laurent, *Understanding Open Source and Free Software Licensing*. Cambridge, Massachusetts: O'Reilly, 2004.

[2] J. Gay, Ed., *Free software, Free society: Selected Essays of Richard M. Stallman*. Boston: GNU Press, Free Software Foundation, 2002. [Online]. Available: http://www.gnu.org/philosophy/fsfs/rms-essays.pdf

[3] D. Harhoff, J. Henkel, and E. von Hippel, "Profiting from voluntary information spillovers: how users benefit by freely revealing their innovations," *Research Policy*, vol. 32, no. 10, pp. 1753–1769, Dec. 2003.

[4] B. W. Carver, "Share and share alike: understanding and enforcing open source and free software licenses," *Berkely Technology Law Journal*, vol. 20, no. 1, pp. 443–481, 2005.

[5] L. Lawrence, *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*. New York: Penguin Group (USA) Inc., 2004.

[6] C. DiBona, S. Ockman, and M. Stone, *Open Sources: Voices from the Open Source Revolution*. O'Reilly, 1999.

[7] D. Spiller and T. Wichmann, "FLOSS final report part 3. FLOSS free/libre open source software: survey and study. Basics of open source software markets and business models," Berlecon Research, Tech. Rep. IST-2000-4.1.1, Jul. 2002. [Online]. Available: http://www.berlecon.de/research/en/reports.php?we_objectID=70

[8] D. A. Wheeler, "The free-libre/open source software (floss) license slide," Sep. 2007. [Online]. Available: http://www.dwheeler.com/essays/floss-license-slide.pdf

[9] J. Lerner and J. Tirole, "The scope of open source licensing," *The Journal of Law, Economics and Organization*, vol. 21, no. 1, pp. 20–56, 2005.

[10] R. Sen, C. Subramaniam, and M. L. Nelson, "Determinants of the choice of open source software license," *Journal of Management Information Systems*, vol. 25, no. 3, pp. 207–239, 2009.

[11] A. Engelfriet, "Choosing an open source license," *IEEE Softw.*, vol. 27, no. 1, pp. 48–49, Jan/Feb 2010.

[12] R. Goldman and R. P. Gabriel, *Innovation Happens Elsewhere: Open Source as a Business Strategy*. San Francisco: Morgan Kaufmann, Elsevier, Apr. 2005.

[13] M.-W. Wu and Y.-D. Lin, "Open source software development: an overview," *IEEE Computer*, vol. 34, no. 6, pp. 33–38, Jun. 2001.

[14] B. Fitzgerald, "The transformation of open source software," *MIS Quarterly*, vol. 30, no. 3, pp. 587–598, Sep. 2006.

[15] B. Kogut and A. Metiu, "Open-source software development and distributed innovation," *Oxford Review of Economic Policy*, vol. 17, no. 2, pp. 248–264, 2001.

[16] T. R. Madanmohan and R. De', "Open source reuse in commercial firms," *IEEE Softw.*, vol. 21, no. 6, pp. 62–69, Nov/Dec 2004.

[17] A. W. Brown and G. Booch, "Reusing open source software and practices: The impact of open source on commercial vendors," in *Software Reuse: Methods, Techniques, and Tools : 7th International Conference, ICSR-7, Austin, TX*, ser. Lecture Notes in Computer Science, C. Gacek, Ed. Springer, 2002, no. 2319/2002, pp. 123–136.

[18] F. Hecker, "Setting up shop: the business of open-source software," *IEEE Softw.*, vol. 16, no. 1, pp. 45–51, Jan/Feb 1999.

[19] M. Ruffin and C. Ebert, "Using open source software in product development: a primer," *IEEE Softw.*, vol. 21, no. 1, pp. 82–86, 2004.

[20] M. K. McKusick and G. V. Neville-Neil, *The Design and Implementation of the FreeBSD Operating System*. Reading, MA: Addison-Wesley, 2004.

[21] E. R. Gansner and S. C. North, "An open graph visualization system and its applications to software engineering," *Softw. Pract. Exper.*, vol. 30, no. 11, pp. 1203–1233, 2000.

[22] R. S. Boyer and J. S. Moore, "A fast string searching algorithm," *Commun. ACM*, vol. 20, no. 10, pp. 262–272, Oct. 1977.

[23] D. M. German, "Using software distributions to understand the relationship among free and open source software projects," in *MSR'07: Proceedings of the Fourth International Workshop on Mining Software Repositories*. Washington, DC, USA: IEEE Computer Society, 2007, p. 24.

[24] D. M. German, Y. Manabe, and K. Inoue, "A sentence-matching method for automatic license identification of source code files," 2010, accepted for publication at Automated Software Engineering. [Online]. Available: http://turingmachine.org/~dmg/papers/dmg2010ninka.pdf

[9]http://sourceforge.net/projects/oslc/

[10]http://mac.softpedia.com/get/Utilities/ASLA.shtml

[11]http://fossology.org/