An empirical study of the reuse of software licensed under the GNU General Public License

Daniel M. German and Jesús M. González-Barahona

Abstract Software licensing is a complex issue in free and open source software (FOSS), specially when it involves the redistribution of derived works. The creation of derivative works created from components with different FOSS licenses poses complex challenges, particularly when one of the components is licensed under the terms of one of the versions of the GNU General Public License (GPL). This paper describes an empirical study of the manner in which GPLed licensed software is combined with components under different FOSS licenses. We have discovered that FOSS software developers have found interesting methods to create derivative works with GPLed software that legally circumvent the apparent restrictions of the GPL. In this paper we document these methods and show that FOSS licenses interact in complex and unexpected ways. In most of these cases the goal of the developers (both licensors and licensees) is to further increase the commons of FOSS.

1 Introduction

Most software today is too complex to be created from scratch. One of the most interesting aspects of free and open source software (FOSS) is the possibility of reusing, without a fee, third party products to produce new ones. From a legal point of view, those new products are derivative (also called derived) works from each of their third-parties components, and their creation and redistribution is restricted by copyright law. FOSS licenses have been created to foster derivative works. Clause 3 of the Open Source Definition states that a open source license should allow the creation and distribution of derived works and their further distribution under the

Daniel M. German
University of Victoria, Canada, e-mail: dmg@uvic.ca

Jesús M. González-Barahona
Universidad Rey Juan Carlos, Spain, e-mail: jgb@computer.org

same license as the original software [25]. Freedom 3 of the The Free Software Definition [3] also grants a similar right: "The freedom to improve the program".

By definition, any FOSS allows the creation of other works based upon the software. Unfortunately it is not always possible to redistribute the resulting software under same license under which each component is made available, hindering the ability of a developer to reuse such component.

FOSS is a healthy software ecology where thousands of different software products interact among themselves to satisfy the user requirements. For example, the Debian 4.0 Linux-based system is composed of more than 18,000 different installable software packages [21] (a software package, in FOSS distributions, corresponds to independent and identifiable software that can be installed in a system and satisfies a particular set of requirements; they range from end-user applications, such as OpenOffice, to highly specialized libraries, such as libtiff, or libjpeg). FOSS packages tend to reuse many different other software packages. We can therefore see each software package as a component. A FOSS application is typically a collection of components that interact among themselves (even if this is not immediately obvious to the user).

Such complex interrelations demonstrate an important feature of FOSS systems: that FOSS is commonly created by reusing other FOSS[1]. As FOSS continues to evolve, the number of FOSS applications, and the median number of packages required by each, keeps increasing—see [21] for a discussion of this phenomenon.

There is, however, an important challenge that needs to be addressed before a software application can reuse the desired component: licensing terms. Anybody interested in reusing a FOSS (we will use the term integrator to describe such individual; integrators might want to create FOSS or proprietary software) should be aware and understand the licensing terms of such component, and how these terms affect if (and how) the component can be used, particularly when components of different licenses are used.

Licenses are restrictions on the way software can be used and combined. The number of FOSS licenses keeps growing. At our last count there were 70 FOSS licenses approved by the Open Source Initiative (OSI) and many more in use that are been approved. This has lead to the problem known as License Proliferation [2].

Some of the most important FOSS licenses are the three versions of the GNU General Public License (GPL). It is widely accepted that any software that incorporates GPLed licensed software (or simply GPLed software) should also be licensed under the GPL (this is the basis for the notion of Copyleft, see [24]).

While investigating the relationships between different packages in software distributions [11, 10, 21] we observed that GPLed software is being used as a required components in applications that are not released under the GPL. For example, any PHP program that connects to a MySQL database requires to use the PHP run-time engine and the MySQL Client Library. The PHP run-time engine uses the MySQL Client Library via linking (either dynamic or static). PHP is licensed under the PHP License 3.0 and the MySQL Client library under the GPL version 2.0 (GPLv2), in

[1] In Debian 4.0 there are only a handful of packages that do not require any other package.

apparent contradiction to the conditions of the GPL, which state that, any program linking to a GPLv2 licensed library should be licensed under the GPLv2 also. In other words, if the PHP run-time engine links to the MySQL Client Library, it can only be licensed under the GPLv2. These observations prompted our main research question: how is GPLed software interacting with software from other licenses?

In this paper we studied 124 FOSS packages—45 were licensed under the terms of the GPL—and how the GPLed packages interacted with non-GPLed ones.

Our results highlight three major results: first, an important concern of the developers of GPLed on having their software used by other FOSS, even under other FOSS licenses; second, the creative manners that some developers use to allow the integration of GPLed components into their products without violating the terms of the license; and third, the strong protections that the GPL offers to some companies who have chosen to release their software under the GPL.

This paper is divided as follows. Section 2 describes the legal issues surrounding FOSS licensing and derivative works. Section 3 describes the different versions of the GNU Public License and how it interacts with other FOSS licenses in the creation of derivative works. We continue in Section 4 with a description of our methodology and data used for the elaboration of this paper. Section 5 summarizes our main results. We end with conclusions and future work.

2 Derivative works and licensing

Copyright law grants the copyright owner of a software product exclusive rights on most aspects related to its modification, distribution and exploitation. Using a license, the copyright owner can also grant specific permissions to third parties to modify, distribute, and in general exploit the software (see [1, 12, 13, 18] for comprehensive discussions on how copyright protects software).

An important exclusive right that can be licensed is the creation and redistribution of derivative works (also known as derived works). In the United States of America, derivative works are defined as “a work based upon one or more preexisting works [...] in which a work may be recast, transformed, or adapted” [27]. Similar definitions can be found in other jurisdictions.

When combining FOSS components with different licenses the concept of license compatibility arises. Assume two components A and B are combined in the creation of a product C (C is a derivative work of both A and B). If it is possible to satisfy in the license of product C—simultaneously—the conditions imposed by the license of A and the conditions imposed by the license of B, then the license of A is said to be compatible with the license of B (and vice-versa). In other words, a product which is the result of integrating some components, and can be considered as a derived work of them, can be redistributed only if the licenses of such components are compatible. Some licenses (including the GPL) require that the license of the derivative work be the same as the license of the component.

For example, as we previously described, the PHP run-time engine reuses the MySQL Connect Library. The PHP run-time is created by combining several components, such as the PHP interpreter, which is licensed under the PHP License 3, and the MySQL Connect Library, which is licensed under the GPL version 2. The license of the PHP run-time should therefore be compatible with the terms of each of its components (in this case the PHP License 3 and the GPL version 2). If the license of the run-time is PHP License 3 then it satisfies the condition of its interpreter license (also PHP License 3), but it does not satisfy the conditions of the GPL version 2 (that the derivative work should be also GPL version 2). Therefore we can say that the PHP License 3 is not compatible with the terms of the GPL version 2 (and vice-versa). We will later discuss how this incompatibility is addressed by the copyright owners of the MySQL Connect library.

A complementary example is Apple's Safari Web browser that is licensed under a proprietary license. It contains several components under various FOSS licenses, such as the GNU Library General Public License version 2 and the new BSD license[2]. In this case the license of Safari satisfies the conditions of all the licenses it uses, hence the GNU Library General Public License version 2 is compatible with the new BSD license (and vice-versa).

The designer of a new product must determine, in advance, not only if the licenses of the components are compatible, but also if they are compatible with the desired licensing terms of the new product. In the proprietary world this is usually an issue of negotiating the licensing terms of each component with its copyright owner. When the component is FOSS this is not easy: frequently an open source component is only available under one license.

To complicate the issue further, depending on how components are mixed, the resulting software can be considered a derivative work or not. For instance, if two programs communicate through sockets or using exec/fork execution, the resulting system is usually not considered to be a derivative work of them. In the other end of the spectrum, if two components are mixed as source code, and compiled together, the resulting software is clearly considered a derivative work[14]. In other scenarios between these two extremes, the situation is less clear, and may have to be settled in court. Rosen considers the question "what is a derivative work of software?" the most difficult legal question facing the FOSS community[22].

To simplify our discussion, for the rest of this paper we will assume that a software system is the derivative work of the components it uses.

3 The GNU General Public License family of licenses

The Free Software Foundation has authored several licenses and can be grouped into three different families: the GNU General Public License (GPL), the GNU Lesser

[2] The original BSD license contains four clauses; one of them, known as the "advertising" clause has been dropped, resulting in what is known as the new BSD license, also known as 3-clauses BSD.

General Public License (LGPL, and previously known as the GNU Library General Public License) and the GNU Affero General Public License (AGPL), as shown in table 1.

Family of Licenses	Released			
	1989	1991	1999	2007
GNU General Public License (GPL)	Version 1	Version 2		Version 3
GNU Lesser General Public License (LPGL)		Version 2	Version 2.1	Version 3
GNU Affero Public License (AGPL)				Version 3

Table 1 The families of GNU licenses for software. There was no version 1 of the LGPL. Version 2 of the LGPL was known as the Library General Public License, but it was renamed to Lesser General Public License in version 2.1. There were no versions 1 or 2 of the GNU Affero General Public License, but there exist version 1 and 2 of the Affero Public License, which were not authored by the Free Software Foundation but are considered as predecessors of the AGPL.

In this paper we have concentrated on the GNU General Public License family, which comprises three licenses (see Table 2): the original GNU GPL (GPL version 1 [4] or GPLv1, published in 1989), version 2 (GPLv2 [5], published in 1991), and version 3 (GPLv3 [6], published in 2007). All of them have similar goals, and each one is derived from its predecessors. However, each license in this family has its own sets of definitions, grants and conditions. From a legal point of view each version is independent of each other, and their effects are different.

Version	**Date**	**Description**
1	Jan 1989	First copyleft license
2	June 1991	Adds 'liberty-or-death' clause: if a user is prevented from satisfying the terms of the license, then she cannot distribute the software
3	June 2007	Clarifies legal definitions, addresses patents, license proliferation, and hardwared-based restrictions

Table 2 The three versions of the GNU General Public License.

Only the GPLv2 and GPLv3 have been approved by the Open Source Initiative as open source licenses[20]. The Free Software Foundation considers the GPLv1 and GPLv2 as deprecated, and recommends licensing under the GPLv3. Nonetheless, a large amount of software is still licensed under GPL v2. On the other hand, software released under GPL v1 is rare.

With respect to derivative works, one of the most important characteristics of GPL licenses is their reciprocity: they allow the redistribution of derivative works of a GPL-licensed software if and only if the derivative work is distributed under the same GPL license. In particular, if a component is licensed under a version of the GPL, then any of its derivative works should also be licensed under the same version of the GPL. Therefore, license compatibility of the GPL can be stated as: a given license L is compatible with a version of the GPL v if a derivative work of two com-

License	GPLv1	GPLv2	GPLv3
GPLv1	Yes		
GPLv2		Yes	
GPLv3			Yes
X11/MIT	Yes	Yes	Yes
Old BSD (4 clauses)			
New BSD (3 clauses)	Yes	Yes	Yes
Apache v1			
Apache v1.1			
Apache v2			Yes
Artistic v1.0			
Artistic v2.0	Yes	Yes	Yes
Common Public v1			
Eclipse Public v1.0			
Mozilla Public v1			
Mozilla Public v1.1			

Table 3 Compatibility of several FOSS licences with the GPL versions. Empty intersections correspond to No compatibility. Note that the any GPL version is not compatible with other versions (table based on information provided by the FSF [7]).

ponents, one distributed under L, and the other under v can be redistributed under v. Table 3 shows some of the most widely used FOSS licenses and their compatibility with the versions of the GPL.

4 Data and Methodology

For this paper our goal was to study how the licenses of different packages are combined in the creation of more complex software packages. Package management systems, such as `apt`, `yum` and `fink` simplify the installation of FOSS; when a user wants a package installed, dependencies are analyzed and the packaging system determines what packages are required, and installs them if necessary.

These packaging systems maintain dependency information for a large number of FOSS. For example, Debian 4.0 contains over 18,000 different packages, and Debian developers maintain the dependency trees of each of them. We selected 8 widely known applications (Apache httpd 2.2, GIMP 2.0, MySQL 5.0.38, KOffice 1.6.3, GNOME Desktop 2.14.0, GCC 4.3.2, PostgreSQL 8.2.4, and Bugzilla 3.0.2) and expanded their dependency trees. The union of these 8 dependency trees resulted in 124 different software packages (for a detailed description of how these trees are computed see [11]). We downloaded the source code of each of these packages and manually inspected their licensing terms. Determining the licensing terms of a package could be expected to be a straightforward task, but we found this not to be the case. Sometimes the licensing terms are clearly stated, but sometimes they are not obvious. Another complication is that in some cases the licensing terms are not simple. For example, the licensing terms of `netpbm` is a file that lists every

single source code file and its corresponding license. In other cases different parts of a package are licensed under different licenses; for example, the run-time library of the `GCC` compiler is "LGPLv3 or a any newer version" (we denote this type of licensing with a '+' symbol after the version number, LGPLv3+) while the rest of the compiler is GPLv3+. In other cases the software is available under more than one license, being the licensee the one who chooses which one applies. For example, `Firefox` is licensed under the terms of the Mozilla Public License version 1.1, the GPL v2+ and the LGPL v2.1+.

We found that 45 packages (36 percent of all inspected packages) were GPLed, while many others were LGPLed. Table 4 shows the number of packages that use each of the GNU licenses. We show these numbers for information purposes only. These results should not be considered an estimation of the frequency with which each license is being used in FOSS, that was not the purpose of our methodology.

GNU License	**Version**	**Freq.**
General Public License (GPL)	1+	1
	2	12
	2+	31
	3+	1
Library General Public License (LGPL)	2	4
	2+	37
Lesser General Public License (LGPL)	2.1	4
	2.1+	11
	3+	1

Table 4 Number of packages using the GNU Licenses (out of the 124 studied packages). The first column lists the number of packages using it. A + after a version number of a license means that the licensor allows the licensee to choose a newer version of the license. Some packages were licensed under the terms of two or more licenses.

With the licensing information we could determine if the licenses of a package and those it directly requires were incompatible. For example, `Bugzilla` is licensed under the Mozilla Public License v1.1, yet it requires `MySQL`, released under the terms of the GPLv2. We assumed that the use of such package was according to the terms of its license (i.e. that `Bugzilla` uses `MySQL` following the terms of the GPLv2). For these cases we tried to determine if one package was considered a derivative work of the other. If it appeared to be, we tried to understand and document the rationale that allowed such use (in apparent contradiction to the terms of the GPL). We inspected their documentation, Web sites and, in some cases, emailed their authors for further clarification. We documented these cases, which are presented in the rest of this paper.

5 Reusing GPLed software

In this section we present a survey of methods used by licensors and licensees of GPLed software to allow the creation of works with components under different, and potentially incompatible licenses. We organize these methods based according to their goals, and exemplify their use. Table 5 offers a summary of all of them.

Method	Examples
Making it "compatible"	MySQL Client library, QT
Dual or multi licensing	Perl, Mozilla, Firefox, Thunderbird
Relicensing under newer versions	Many packages
"Licensed as"	Perl modules
Clarification of terms	Linux, Perl
Linking to non-compatible licenses	OpenSSL, Sun's Java JDK

Table 5 Methods used when releasing software packages that include components under GPL licenses and some others under licenses apparently incompatible with them.

5.1 Making the GPL "compatible" with other FOSS licenses

Sometimes the copyright owner of a library wants to maintaing it licensed under the GPL. But at the same time, she wants to allow the library to be linked by software under other FOSS licenses, including those incompatible with the GPL.

5.1.1 Example: MySQL Client libraries

MySQL AB originally licensed the MySQL Client libraries under the terms of the LGPLv2. The LGPL allows the creation and distribution of software under any license to link to the library (subject to some conditions). In 2004 MySQL AB decided to release new versions of the libraries under the GPLv2 instead. Suddenly, many applications under licenses not compatible with the GPL were not allowed to link to them any more (for example PHP-based applications).

MySQL AB realized that they wanted to continue allowing some of these applications to link to the library but did not want to release the libraries under several FOSS licenses. MySQL AB addressed this problem by issuing the "MySQL FLOSS License Exception" [15]. The company explains its rationale as follows [16]:

> We want specified Free/Libre and Open Source Software ("FLOSS") applications to be able to use specified GPL-licensed MySQL client libraries (the "Program") despite the fact that not all FLOSS licenses are compatible with version 2 of the GNU General Public License (the "GPL").

The exception is an addendum to the GPLv2, and places two main conditions on creating derivative works that link to the library and are not released under the GPLv2, which can be summarized as follows:

1. If the library is modified, then such changes should be released under the GPL;
2. the rest of the derivative work is released under one of 24 different licenses listed in the exception (which includes the BSD, MIT, Mozilla Public v1.0 and v1.1, Apple Public Source v2, PHP, Python Software Foundation v2.1.1, LGPLv2, and v2.1, and the Apache v1.0, v1.1, and v2.0).

5.1.2 Example: Qt

Qt is a cross-platform GUI library. In Sept 2000 Trolltech decided to release new versions of it under the GPLv2. Unfortunately, this made Qt incompatible with more permissive licenses, such as the LGPL (some LPGLed software was already linking to Qt). To solve this problem, Trolltech issued an exception to the GPLv2, known as the Nokia Corporation Qt GPL Exception Version 1.3 [19] (Nokia bought Trolltech in 2008 and changed the name of the exception from Trolltech Qt GPL Exception). It states the following conditions:

1. Software under one of the listed licenses can link to the library;
2. A commercial application can link to a pre-installed version of the library if it was developed "in accordance with the terms and conditions of the Qt Commercial License Agreement."

This exception lists 31 licenses (compared to 24 of the *MySQL* exception). The addition of the LGPLv3 and the Eclipse Public v1.0 are the most significant differences. In January of 2009 Nokia changed the license of QT to LGPLv2.1, making this exception unnecessary.

The *MySQL AB FLOSS License Exception* and the *Nokia Corporation Qt GPL Exception* have very similar objectives, yet they are drafted in very different terms. Nokia's exception permits linking with components under certain licenses, while MySQL's defines what constitutes a derivative work and issues exceptions for some FOSS licenses.

5.2 Letting the licensee choose: dual and multi licensing

The concept of dual, and more generally speaking, multi-licensing (the term *disjunctive* licenses is also used). refers to the method of licensing in which the licensor gives a choice to the licensee to select from two or more licenses. It has been usually associated with a business model in which a company makes the software available under a FOSS license, and a commercial license (see [28] for a discussion of this licensing method and its benefits). Multi-licensing has also become a method to address license incompatibility.

5.2.1 Example: Perl

The licensing terms of `Perl` allow the recipient of the software to choose the license under which to modify and redistribute the software. The choice is between GPL v1 or later and the original Artistic License (incompatible with the GPL).

5.2.2 Example: Mozilla, Firefox, and Thunderbird

The Mozilla Foundation makes `Mozilla`, `Firefox` and `Thunderbird` available under three different licenses: the Mozilla Public License (MPL) version 1.1, the GPLv2 or later, or the LGPL v2.1 or later, at the choice of the licensee.

5.3 Relicensing under "any newer version"

As shown in table 4, many licensors of GPLed software give the opportunity to the licensee to choose among a GPL license or "any newer version of the license". Many FOSS packages use this method of licensing. In fact, this practice is recommended by the Free Software Foundation, and used in all GNU software. The main rationale for it is to automatically adapt to future versions of the GPL (as it happened when the GPL v3 was released in 2007).

5.3.1 Example: Perl

`Perl` allows its licensee to use the GPL v1 or any newer version of the license. It follows the practice that the Free Software Foundation recommends in the Appendix of the GPL itself, which explains that the following text should be added at the beginning of all source code files of a software package:

> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 1, or (at your option) any later version.

The use of such wording implies that the licensor is giving the option to the licensee to choose another license (the newer version). The rationale given by the Free Software Foundation is that "makes it possible for us to change the distribution terms on the entire collection of GNU software, when we update the GPL" [8]. The most clear advantage is that the licensee does not have to worry about tracking updates to the GPL, and does not have to worry about relicensing the software package under every new version of the GPL[3].

From a legal point of view this appears to be no different from *multi-licensing* except that the newer license might not be yet written at the time the software is

[3] When multiple authors are involved, relicensing might be difficult, or even impossible.

initially released. This method of licensing requires trust in the Free Software Foundation who is responsible for drafting the new versions of the GPL, as the new versions could be more or less restrictive than the current one.

5.4 "Licensed as ... " to allow relicensing

Sometimes software is licensed indirectly, by stating that its license is the same as another one (and not stating a license explicitly). In this case one component explicitly states its license and the rest indicate they are licensed in the same way as such component. If such component changes its license, all the other components immediately change license[4]. This simplifies the licensing (and potential future relicensing) of components that are likely to interact between themselves.

5.4.1 Example: Perl modules

The Perl modules `libtemplate`, `libmailtools`, `libmime-tools-perl`, and many others, are licensed as:

> This module is free software; you can redistribute it and/or modify it under the same terms as Perl itself.

In other words, the copyright owners of each of these packages have given the copyright owners of Perl (the Perl Foundation[5]) the ability to choose the license of such packages. This method of licensing simplifies the distribution of these modules along Perl now and in the future.

5.5 Clarifying the terms

Licenses are legal documents, and despite trying to be as precise as possible, in some cases the terminology may be ambiguous to those who need to apply it in a given scenario. In the case of the GPL licenses, perhaps the most contentious issue is what constitutes a derivative work. The Free Software Foundation has a FAQ for this purpose, which argues that any program that links (dynamically or statically) to a GPL library should be considered a derivative work of the library [8].

Only the copyright owner can take a copyright license violation to court. It is therefore valuable to know how they interpret any ambiguous section of the license.

---

[4] Because the component does not have a specific license itself, this method poses difficulties for those tracking the licenses of software, such as `sourceforge.net`.

[5] `http://www.perlfoundation.org`

5.5.1 Example: Linux GPL clarification

Linus Torvalds (the main copyright owner of the Linux kernel) has stated that programs that only use the services of the kernel are not considered derivative works of the kernel [26]. Without this clarification some feared that any program that runs under Linux would have to be licensed under the GPLv2, as Linux is licensed.

5.5.2 Example: Perl GPL clarification.

Larry Wall, the original author and major copyright owner of Perl includes a clarification to Perl's licensing terms. It states "my interpretation of the GNU General Public License is that no Perl script falls under the terms of the GPL unless you explicitly put said script under the terms of the GPL yourself." [29]. Otherwise some feared that any Perl script would be a derivative work of the Perl interpreter (when the script is executed it invokes functions in the interpreter).

5.6 Linking to code with an incompatible license

Sometimes there exists a library that GPLed software would like to link to, but its license has one ore more clauses that are incompatible with the GPL. In some cases it is possible to do so. Two remarkable cases are the OpenSSL exception and the JDK CLASSPATH exception.

5.6.1 Example: The OpenSSL exception

A very interesting case surrounding the issue of license compatibility involves the OpenSSL library. OpenSSL is a cryptographic implementation of the SSL and TLS protocols, being FIPS 140-2 compliant (an important requirement for certain organizations which use cryptographic software). All of this makes it desirable for many applications to link to it [17]. OpenSSL is released under the terms of both the OpenSSL License and the SSLeay License. These licenses are incompatible with the GPL licenses due to the requirement to acknowledge the name of the library (these type of GPL-incompatible clauses are known as advertising clauses).

As a workaround, developers of GPLed software interested in linking to the OpenSSL library have added an exception clause to the GPL:

> In addition, as a special exception, the copyright holders give permission to link the code of portions of this program with the OpenSSL library under certain conditions [...] You must obey the GNU General Public License in all respects for all of the code used other than OpenSSL. If you modify file(s) with this exception, you may extend this exception to your version of the file(s), but you are not obligated to do so. [...]

The program linking to the OpenSSL is still obliged to satisfy the conditions of the OpenSSL license, but only for the OpenSSL code. For the rest the GPL license applies.

5.6.2 Example: JDK and the CLASSPATH exception

Until recently Sun distributed its Java JDK under the Common Development and Distribution License (CDDL), an OSI approved license that is not compatible with any of the GPL licenses. Sun considered to change the license of the JDK to the GPLv2, but there was a major roadblock: any program that runs under the Java Virtual Machine (JVM) dynamically links to the runtime library (and to any library found in the directories listed in the CLASSPATH environment variable). The runtime library is part of the JDK, and would be licensed under the GPLv2 too. As a consequence any program running under the JVM would need to be licensed under the GPLv2. To avoid this issue Sun added the CLASSPATH exception to its licensing terms for Java. This exception, authored by the Free Software Foundation, explicitly states that linking to the runtime library (and other libraries located in the CLASSPATH) is not considered a derivative work[9, 23]:

> As a special exception, the copyright holders of this library give you permission to link this library with independent modules to produce an executable, regardless of the license terms of these independent modules, and to copy and distribute the resulting executable under terms of your choice [...] If you modify this library, you may extend this exception to your version of the library, but you are not obligated to do so. [...]

6 Discussion, Conclusions and Further work

We do not expect this list to be comprehensive. It is likely that other innovative methods are used to combined software of incompatible licenses with the GPL.

Another important issue is that the methods herein presented have not been tested in courts of law. We believe, however, that many of them have been drafted by intellectual property lawyers who are probably well versed in the legal ramifications of such methods (e.g. it is likely that IP lawyers for Nokia, MySQL AB, Sun, the Free Software Foundation and the Mozilla Foundation have drafted and approved the methods used by their corresponding organizations).

The license compatibility and license proliferation problems are important for FOSS developers. The Free Software Foundation and the copyright owners of some licenses have been trying to curve this problem by issuing new versions of their licenses that make them compatible with the GPLv3. For example, the Artistic License v2.0 and Apache License v2 are both compatible with the GPLv3.

In this paper we have studied and classified ways in which software released under GPL licenses can be combined with components under other licences incom-

patible with it. This study shows that many copyright owners who want or need to use GPLed licensed software have to resolve GPL incompatibility issues.

In FOSS, the topic of license incompatibility is a complex, yet very important one. Copying and reusing of code and combining FOSS components are two common practices in this community. But the resulting works could be impossible to distribute if the incompatibility issues are not properly addressed. In many cases, if the copyright owners take the appropriate measures, this problem can be alleviated, and even managed in relatively simple ways.

License incompatibility is related to another major problem for the FOSS community: license proliferation. As more and more FOSS licenses are used, the chances of license incompatibility when producing a compound work are larger and larger, thus limiting the benefits of code reuse. A study of the relationship between both issues is therefore of great practical interest for FOSS developers.

Knowing how many cases are affected (or potentially affected) by license incompatibility is not a simple task. Tools that help to determine the licenses involved in a derivative work (by finding the licenses of the components and dependencies), and identify potential incompatibility issues could help in this realm. Quantitative analysis of number of packages affected, for instance, in GNU/Linux distributions, could also shed some light on the matter.

Acknowledgments

The work of Daniel M. German has been funded by Hewlett-Packard to support the FOSSology Project. The work of Jesus M. Gonzalez-Barahona has been funded in part by the European Commission, under the FLOSSMETRICS project (FP6-IST-5-033547) and by the Spanish CICyT, under the SobreSalto project (TIN2007-66172). We want to thank the anonymous reviewers for their helpful comments.

References

1. Andrew Becerman-Rodau. Protecting Computer Software: after Apple Computer Inc. v. Frankin Computer Corp., 714 F.2d 1240 (3d Cir. 1983) does copyright provide the best protection? *Temple Law Review*, 57(527), 1984.
2. Ken Coar. The licence proliferation project. Open Source Initiative, `http://www.opensource.org/proliferation`, July 2006.
3. Free Software Foundation. The free software definition. `http://www.gnu.org/philosophy/free-sw.html`.
4. Free Software Foundation. Gnu general public license version 1. `http://www.gnu.org/licenses/old-licenses/gpl-1.0.txt`. Accessed Nov. 2008.
5. Free Software Foundation. Gnu general public license version 2. `http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt`. Accessed Nov. 2008.
6. Free Software Foundation. Gnu general public license version 3. `http://www.fsf.org/licensing/licenses/gpl.html`. Accessed Nov. 2008.

7. Free Software Foundation. Licenses. http://www.fsf.org/licensing/licenses/, 2008. Accessed Nov. 2008.
8. Free Software Foundation. Frequently Asked Questions about the GNU Licenses. http://www.fsf.org/licensing/licenses/gpl-faq.html. Accessed Noc. 2008.
9. Free Software Foundation. GNU Classpath. http://www.gnu.org/software/classpath/license.html, 2008. Accessed Sept. 2008.
10. Daniel M. German. Using software distributions to understand the relationship among free and open source software projects. In *4th International Workshop on Mining Software Repositories (MSR 2006)*, May 2007.
11. Daniel M. German, Jesús M. González-Barahona, and Gregorio Robles. A model to understand the building and running inter-dependencies of software. In *"Proc. 14th Working Conference on Reverse Engineering"*, pages 130–139, 2007.
12. Paul Goldstein. *International Copyright: Principles, Law, and Practice*. Oxford University Press US, 2001.
13. Stanley Lai. *The Copyright Protection of Computer Software in the United Kingdom*. Hart Publishing, 2000.
14. Nancy J. Mertzel. Copying 0.03 percent of software code base not "de minimis". *Journal of Intellectual Property Law & Practice*, 9(3):547–548, 2008.
15. MySQL AB. MySQL AB FLOSS License Exception. http://www.mysql.com/company/legal/licensing/foss-exception.html, March 2007. Accessed Dec. 2007.
16. MySQL AB. MySQL 5.0 Reference Manual. http://dev.mysql.com/doc/refman/5.0/en/, 2008.
17. National Institute of Standards and Technology. Validated FIPS 140-1 and FIPS 140-2 Cryptographic Moduples 2007. http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2007.htm.
18. Melville B. Nimmer and David Nimmer. *Nimmer on Copyright*. Matthew Bender & Company, 2002.
19. Nokia. Nokia Corporation Qt GPL Exception Version 1.3. http://doc.trolltech.com/4.4/license-gpl-exceptions.html, 2008. Accessed Nov. 2007.
20. Open Source Initiative. Open Source Licenses. http://www.opensource.org/licenses, accessed Nov. 2008, 2006.
21. Gregorio Robles, Jesus M. Gonzalez-Barahona, Martin Michlmayr, and Juan Jose Amor. Mining large software compilations over time: another perspective of software evolution. In *MSR '06: Proceedings of the 2006 International Workshop on Mining Software Repositories*, pages 3–9, New York, NY, USA, 2006. ACM Press.
22. Lawrence Rosen. *Open Source Licensing: Software Freedom and Intellectual Property Law*. Prentice Hall, 2004.
23. Sun Microsystems. Free and Open Source Java. http://www.sun.com/software/opensource/java/faq.jsp, 2008. Accessed Sept. 2008.
24. The Free Software Foundation. What is Copyleft? http://www.gnu.org/copyleft/, accessed Nov. 2008.
25. The Open Source Initiative. The Open Source Definition. http://opensource.org/docs/osd, 2006.
26. Linus Torvalds. Note to the GNU General Public License. ./COPYING file in the Linux kernel version 2.6.23. Accesed Dec. 2007.
27. United States Copyright Office. Circular 92 Copyright Law of the United States of America and Related Laws Contained in Title 17 of the United States Code, June 2003.
28. Mikko Valimaki. Dual Licensing in Open Source Software Industry. *Systemes d'Information et Management*, 8(1):63–75, 2003.
29. Larry Wall. Perl Kit Version 5. ./README file in Perl version 5.6.10, available at cpan.org. Accesed Dec. 2007.