Analyzing the Relationship between the License of Packages and Their Files in Free and Open Source Software

Yuki Manabe[1], Daniel M. German[2,3], and Katsuro Inoue[3]

[1] Kumamoto University, Japan
[2] University of Victoria, Canada
[3] Osaka University, Japan

Abstract. Free and Open Source Software (FOSS) is widely reused today. To reuse FOSS one must accept the conditions imposed by the software license under which the component is made available. This is complicated by the fact that often FOSS packages contain files from many licenses. In this paper we analyze the source code of packages in the Fedora Core Linux distribution with the goal of discovering the relationship between the license of a source package, and the license of the files it contains. For this purpose we create license inclusion graphs. Our results show that more modern reciprocal licenses such as the General Public License v3 tend to include files of less licenses than its previous versions, and that packages under an Apache License tend to contain only files under the same license.

1 Introduction

One way of reducing software development cost is to reuse components. Today Free and Open Source Software (FOSS) has become a common and viable source of components that are ready to be reused. It is possible to find many components from open source project hosted by open source project hosting site such as SourceForge.net[1], Google Code[2] and GitHub[3]. In addition, users can use software component retrieval system such as SPARS-J[1] and Oholo code[4].

The relationship between licenses is complex. It is not trivial to understand when a file with one license can be reused inside a package of another license. Some authors of licenses provide guidelines that try to clarify this; for example, the Free Software Foundation tries to clarify the relationship between the General Public License and other licenses [2]. However, there is a lack of empirical evidence that shows the relationship between the license of the package and the license of the files it contains.

[1] `sourceforge.net`
[2] `code.google.com`
[3] `github.com`
[4] `ohloh.net`

L. Corral et al. (Eds.): OSS 2014, IFIP AICT 427, pp. 51–60, 2014.
© IFIP International Federation for Information Processing 2014

In this paper we describe an empirical study that investigates how different software licenses are reused as white-box components in the software packages found in Fedora, a popular Linux distribution. We show the relationships between the licenses of packages, and the licenses of its files. Our goal is to assist developers, license compliance officers and lawyers in understanding how licenses are actually used.

The contributions of this paper are:

- An empirical study of how the relationship between the license of a package and the license of the files it contains (white-box reuse).
- The definition of *License Inclusion Graphs* that show the licenses that are used inside a packages of a given license.
- The license inclusion graphs from Fedora version 19 for the most popular licenses.

2 Background

The monetary cost of reusing a FOSS component can be zero, but it requires the user to read and accept its FOSS license. In general a software license is a set of permissions that the intellectual property owner grants to the user of the software after a set of conditions have been satisfied (these conditions could be, for example, the payment of a fee). A FOSS license is a software license where the permissions granted include the right to make derivative works of the software and further redistribute those works in exchange to the acceptance of certain conditions (see [3] for a detail description of FOSS licenses, and [4] for a formalization of grants and conditions). For example, the original X11/MIT license grants permissions to "deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so"; the only condition it places to grant those rights is "The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software."

When developers built software by reusing FOSS, either by linking to them or by copying their source code, they have to satisfy the conditions of each of their licenses [4]. Fig. 1 illustrates this problem. In this scenario, the main source code of the project is developed under license A. It reuses two libraries by linking to them (reused as components–i.e. black-box reuse) each under license B and C. To complicate things further source code files from another component (under License D) has been copied into the project. The problem becomes, what license can be used to release the new software system that is compatible with the license of the source code (A) the copied source code (D) and the linked components (B and C). This is not always an easy to answer question, especially when the newly created project reuses many components. For example, Scacchi et. al. [5] examine Google Chrome and showed that it uses 27 components and libraries under 14 different licenses.

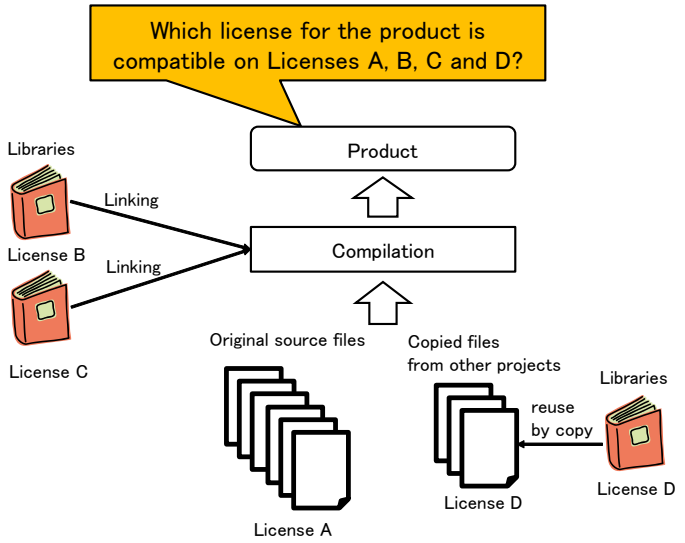

Fig. 1. Example of the challenges of reuse of various licenses. In this example, the code has been developed under license C, but some of the source code was copied from a component with license D. Later this code was compiled and linked with components under license A and B. Can the resulting product be licensed under license A, B, C, D or other license?.

Another problem is the proliferation of licenses. The Open Source Initiative[5], the body responsible for the definition of Open Source, has approved 69 licenses as open source [6] and BlackDuck claims that the Black Duck Knowledge Base includes data related to over 2200 licenses[7]

Previous research has studied license incompatibility. Alspaugh et. al. proposed a model of licensing terms represented by a tuple ⟨actor, modality, action, object, license⟩ and an approach to calculate conflicts of rights and obligations that they implemented on ArchStudio4, a software traceability tool. German et. al. shows integration pattern to solve conflicts of license[4]. However, solving license conflicts is not an easy task since it requires legal background. Also, most of the literature regarding license compatibility has focused on the Free Software Foundation licenses (the different versions of the General Public License —GPL— and the Lesser General Public License –LGPL).

In general, files are expected to contain licensing information in them, embedded in the comments of the file. This licensing information describes one or more licenses under which the file is "open-sourced". For this paper we analyzed as a corpus of FOSS the source packages of the Fedora Core 19 distribution. A FOSS system is usually represented in Fedora as a "source" package. A source package is a collection of files, including source code and documentation from where binaries are created.

[5] `opensource.org`

Table 1. Names of common open source licenses and their abbreviations as used in this article

Abbrev.	Name
Apache	Apache Public License
BSD4	Original BSD, also known as BSD with 4 clauses
BSD3	BSD4 minus advertisement clause
BSD2	BSD3 minus endorsement clause
CPL	Common Public License
CDDLic	Common Development and Distribution License
EPL	Eclipse Public License
GPL	General Public License
LibraryGPL	Library General Public License (also known as LGPL)
LesserGPL	Lesser General Public License (successor of the Library GPL, also known as LGPL)
X11/MIT	Original license of X11 released by the MIT

Fedora documents the license of every source package. We call this the *declared license* of the package. This information is intended to inform the user of the package of the legal obligations acquired when using it. This information is usually created manually by inspecting the documentation of the software [8]. Fedora restricts what licenses packages can have in order to be included in the distribution, but this list is fairly comprehensive, currently including 253 different FOSS licenses[9].

Table 1 shows the most common license names and the abbreviation used in this paper. This paper uses a suffix v<number> as means to specify a version of that license; for example, *GPLv2* means *GPL version 2*. If the license name is followed by "+" then the file can be used under such license or newer versions of it; for example *GPLv2+* means the file can be used under the terms of the *GPLv2* or the *GPLv3* (including future versions of the license).

3 License Inclusion

Any FOSS package is licensed under at least one FOSS license, and each of its source code files is expected to be licensed under a FOSS license too. Ideally every file should explicitly indicate its license, although it is not uncommon to find files without a license.

If a software package is of a given license, it does not mean that all the files that compose it are also of the same license. For example, it is widely accepted that files under the MIT/X11 license is included in software that is licensed under any other license, as long as the minimal requirements of MIT/X11 license are satisfied (see above). We call this relationship the **inclusion** of one license into another license. In this case, it means that the MIT/X11 license is included in packages under the GPLv2. Using the inclusion relationship, we can compute the **licensing inclusion graph** of a given collection of software packages as follows:

- Create a node for each of the licenses of packages.
- Create a node for each license found in files.
- For every file f in a package p, we create a directed edge $license(f) \rightarrow license(p)$.

We then define the **license inclusion graph of a package license** as the subgraph that ends in a given package license. All the edges in the graph which share a single destination node and come from different nodes with the same license are merged into a single edge. We put the number of the merged edges as the *weight* of this new edge.

For example, assume that a package in GPLv3 includes 10 source files in BSD2. This relation is represented as a graph including two nodes "BSD2" and "GPLv3" and a single edge from "BSD2" to "GPLv3" with weight 10.

4 Empirical Study

We conducted an case study of FOSS software packages. Its goal is to answer the question *What are the inclusion relationships between licenses of packages and licenses of source code?*

4.1 Subject

For this study we used the 2484 source packages in Fedora Core 19[6]. Each package includes an archive of source files and one or more spec files. The spec files provide metadata of the source package including its license.

4.2 Methodology

To answer the research question we created the license inclusion graphs of package licenses found in Fedora, using the following procedure: First we extracted for each package its declared license, and the license of its files as follows:

1. We extracted its declared license from its spec file.
2. For each source code file in the package, we identified its license(s). For this step we used Ninka[7], a license identification tool with a reported accuracy of 93% [10]. Only 2,013 packages have at least one source code file. The median number of files per package is 60 files, average 748.5, and maximum 125,400.

Ninka is not able to identify the license of all files. Frequently this is because Ninka does not know the way a specific project licenses its files. In that case, Ninka reports "Unknown". In 62.2% of the packages, at least one file was "Unknown" (i.e. in 37.8% of the packages, Ninka identified the licenses of all the files). The distribution of the proportion of "Unknown" files had median of 2.7%, with a 3rd quartile of 25%. This meant that we had incomplete licensing

[6] `ftp:///ftp.iij.ad.jp/pub/linux/fedora/releases/19/Fedora/source/SRPMS/`
[7] `ninka.turingmachine.org`

information of packages with high ratio of "Unknown". For this reason we decided to remove from the study packages that had a proportion of 50% or more "Unknown" files (328 packages–16% of the total).

Some packages contain more than one spec file. If the licenses of the spec files of a package were different, that meant that some files in the package were distributed under one license, and some under another. For that reason we removed packages with spec files with different licenses. This resulted in another 210 packages being removed from our study.

In total, we kept for our analysis 1,475 packages with at least one source code file, for total of different 511,308 files.

The abbreviation names of licenses in Fedora are different than the names used in Ninka. To make both lists comparable we created an equivalence table. An excerpt of this table is shown in Table 2. The complete table can be found in the replication package (available at `http://www.dbms.cs.kumamoto-u.ac.jp/~y-manabe/replication_package/index.html/`). Table 3 shows the most frequently found declared licenses in packages. Table 4 shows the number most frequently licenses in source files.

With this information we created the license inclusion graphs of the distribution, and from it the license inclusion graph of the most common licenses.

Table 2. Excerpts from conversion table of license names between Fedora and Ninka

Declared license of package (*Fedora Name*)	License in file (*Ninka Name*)
ApacheSoftwareLicense	Apachev2
ASLv2	Apachev2
BSDwithadvertising	BSD4
EPL	EPLv1
LGPLv2	LibraryGPLv2
LGPL2.1	LesserGPLv2.1
PHP	phpLicV3.01

4.3 Results

Due to space limitations we only show the subgraphs of the most common licenses. The rest are available in the replication package. Figure 2 shows the file-to-package licensing subgraphs for the different versions of the General Public License. The number of package in each source file license node means how many packages have files under the license. The sum of the number of packages in source file is often larger than that in package license node because many packages include license not under same license but under various license. As it can be observed, there is no apparent inconsistency: all licenses of the files (the left node) can be relicensed as the license on the packages (the right node). What is apparent is that the GPLv3+ reuses fewer licenses.

Table 3. Number of packages under each license more than 10 packages)

License name	Packages
GPLv2+	338
LGPLv2+	205
X11mit	154
GPL+ or Artistic	109
BSD	91
GPLv3+	63
ASL 2.0	50
GPLv2+ and GFDL	48
GPLv2	44
GPLv2+ and LGPLv2+	30
LGPLv2	28
LGPLv2+ and GPLv2+	19
LGPLv3+	16
EPL	16

Table 4. Most common licenses used by source files

License Name	Src Files
NONE	111311
EPLv1	70004
GPLv2+	48063
Unknown	32874
LibraryGPLv2+	32745
GPLv3+	29041
BSD3	25570
Apachev2	22099
LesserGPLv2.1+	21733
BSD2	19844
X11mit	15365
GPLv2-classPathExcep	14509
GPLv2	13958
CPLv1	12060

Figure 3 shows the subgraph for the most commonly found permissive licenses. The LesserGPLv2+ shows two main inconsistencies: the use of the GPLv2+ and the GPLv3+. We inspected the files that created this inconsistency, and in most cases, they were in directories that contained the name "test" or "demo", suggesting they were testing and sample programs (most of them very small). It is interesting that these packages would be under the LesserGPL, but the test and demo files under the GPL.

Fedora does not distinguish between the BSD2 and the BSD3, labelling both BSD, as shown in the diagram. In both the BSD and the X11/MIT, the proportion of files without a license is higher than other licenses ($> 30\%$). The "GPL+ or Artistic" is a license that is used mainly for Perl and its modules. It is interesting that most of the files in this graph have no license, or the license "License Same as Perl". This is likely because the template for the creation of a module in Perl uses this license. Notice that in this license, as well as the two Apache, no other license is used. These communities (Perl and Apache) do not seem to reuse code under other licenses. Packages under Apache have the simplest graphs: most files ($> 90\%$) contain the given license, and in the case of Apachev2, only 1.5% of files are under another license. For Apachev1.1 all its files are under the same license.

5 Limitations and Threats to Validity

With respect to the threat of internal validity, in this paper we didn't consider how source files were used. We focused on only relations between a source package and source files included in it. However, not all source files in a package are used in building software. Therefore, we may extract the relations between packages and unused source files. We believe this effect is small.

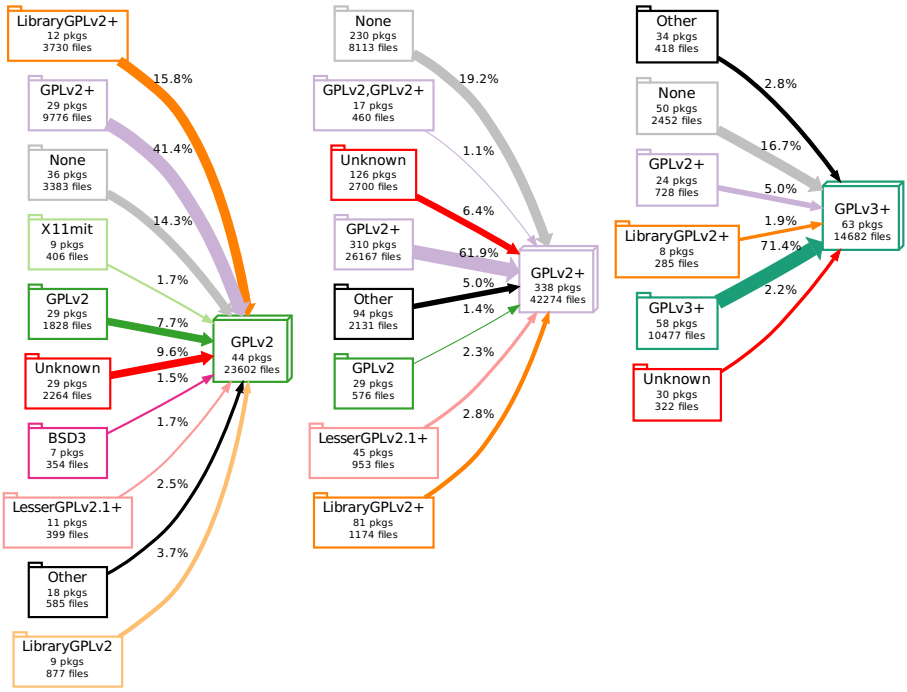

Fig. 2. License inclusion subgraphs for different versions of the GPL

Regarding construction validity, we used Ninka to identify the license of each source file. The quality of the data extracted depends heavily on Ninka's license detection quality. In our previous work [10] we reported that Ninka is accurate in 93% of files. We believe this is enough to represent the most common relationships between licenses. To identify the license of software packages we used the spec files created by the Fedora Project. Our study depends on the accuracy of this information. German et. al. [11] observed that this data is manually generated and an mostly correct. There were very few cases, however, where a package had upgraded to a new license, but the license in the spec file was not updated.

Regarding external validity, we only used source packages in Fedora19. Our results are affected by the selection bias of the Fedora Core Project. To make sure that licenses are comprehensively represented it is necessary to analyze other repositories of FOSS. We plan to do this for future work.

6 Related Work

German et.al[11]. studied license compatibility in software packages by identifying licenses of source packages and source files in them. They found that in general, identifying the license of a package from its source code is not a trivial

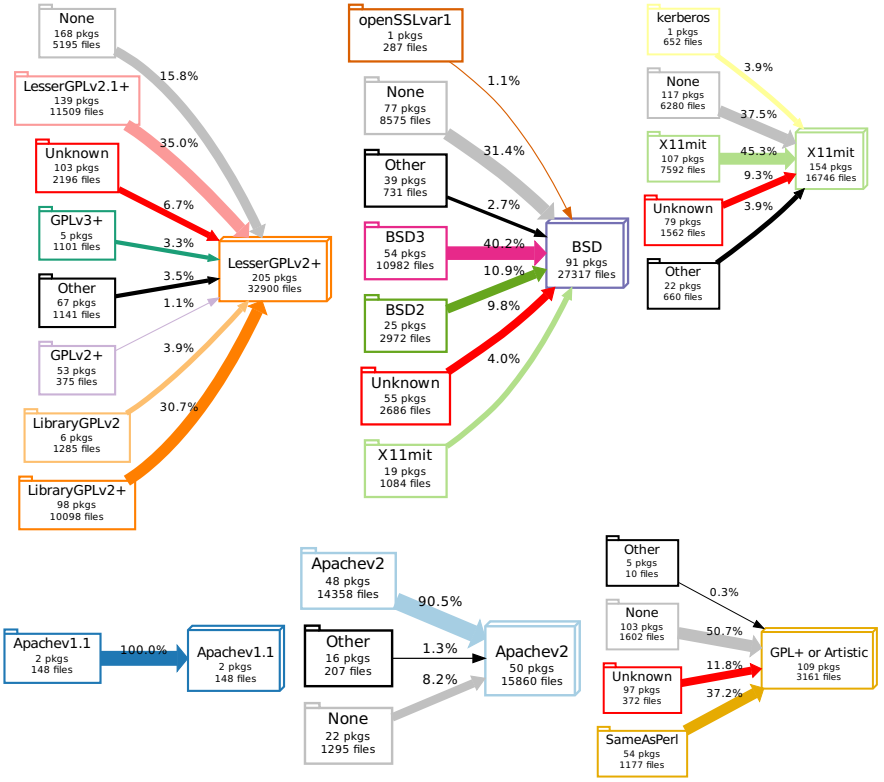

Fig. 3. License inclusion subgraphs of permissive licenses

problem. Our work is an extension to theirs, as we use similar data. The difference is that we focused on the relationships between licenses and not packages.

Stewart et.al [12] addressed the impact of licenses on software projects. Alspaugh et. al [13] have analyzed the requirements that licensing imposes on software. Scacchi et. al [5] have looked into the impact of licenses in the evolution of FOSS software.

7 Conclusions

In this paper we extracted the relationship between the licenses of packages and the licenses of the files are composed of in the Fedora Core 19 distribution. We visualize this information using license inclusion graphs. These graphs show that the different variants of the General Public License are more likely to include other licenses, while licenses such as the Apache tend to contain files only under the same license and they are better at stating the license of their files.

As future work, we would like to analyze the build systems of packages to determine which files are actually part of the binaries. We would like to also

explore the licensing relationships between packages, and repeat this study in other collections of FOSS.

Acknowledgements. This work is supported by Japan Society for the Promotion of Science, Grant-in-Aid for Young Scientists (B) (No.24700029), Grant-in-Aid for Scientic Research (S) Collecting, Analyzing, and Evaluating Software Assets for Effective Reuse (No.25220003), and by an Osaka University International Collaboration Research Grant.

References

1. Inoue, K., Yokomori, R., Yamamoto, T., Matsushita, M., Kusumoto, S.: Ranking significance of software components based on use relations. IEEE Trans. Softw. Eng. 31, 213–225 (2005)
2. Foundation, F.S.: Various licenses and comments about them, `http://www.gnu.org/licenses/license-list.en.html`
3. Rosen, L.: Open Source Licensing: Software Freedom and Intellectual Property Law. Prentice Hall (2004)
4. German, D.M., Hassan, A.E.: License integration patterns: Addressing license mismatches in component-based development. In: Proc. ICSE 2009, pp. 188–198 (2009)
5. Scacchi, W., Alspaugh, T.A.: Understanding the role of licenses and evolution in open architecture software ecosystems. Journal of Systems and Software 85(7), 1479–1494 (2012)
6. Open Source Initiative: Open source licenses, `http://opensource.org/licenses/index.html`
7. Black Duck Software: Black duck knowledge base, `http://www.blackducksoftware.com/products/knowledgebase`
8. Callaway, T.: Fedora: Licensing guidelines (2011), `https://fedoraproject.org/wiki/Packaging:LicensingGuidelines?rd=Packaging/LicensingGuidelines`
9. Callaway, T.S.: Fedora: Software licenses (2013), `https://fedoraproject.org/wiki/Licensing:Main?rd=Licensing#SoftwareLicenses`
10. German, D.M., Manabe, Y., Inoue, K.: A sentence-matching method for automatic license identification of source code files. In: Proc. ASE 2010, pp. 437–446 (2010)
11. German, D.M., Di Penta, M., Davies, J.: Understanding and auditing the licensing of open source software distributions. In: Proc. ICPC 2010, pp. 84–93 (2010)
12. Stewart, K.J., Ammeter, A.P., Maruping, L.M.: Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects. Info. Sys. Research 17, 126–144 (2006)
13. Alspaugh, T., Asuncion, H., Scacchi, W.: Intellectual property rights requirements for heterogeneously-licensed systems. In: Proc. RE 2009, pp. 24–33 (September 2009)