A Preliminary Study on Impact of Software Licenses on Copy-and-Paste Reuse

Yu Kashima
Graduate School of Information Science and Technology, Osaka University
y-kasima@ist.osaka-u.ac.jp

Yasuhiro Hayase
Faculty of Information Sciences and Arts, Toyo University
hayase@toyo.jp

Norihiro Yoshida
Graduate School of Information Science, Nara Institute of Science and Technology
yoshida@is.naist.jp

Yuki Manabe
Graduate School of Information Science and Technology, Osaka University
y-manabe@ist.osaka-u.ac.jp

Katsuro Inoue
Graduate School of Information Science and Technology, Osaka University
inoue@ist.osaka-u.ac.jp

ABSTRACT

Source code of open-source software is permitted to be reused when and only when the conditions of its license are satisfied. There are many different conditions for reusing, since various open-source licenses are used. Therefore, the license of the source code may affect the frequency of reusing or the property of the software for which the source is reused. To identify the relationship between software license and reusing, we are planning to classify copy-and-pasted code fragments based on the license of the fragments. This paper presents a preliminary and manual investigation on a small source file set. The result indicates that the license of a fragment affects the quantity and the license of copied fragments.

Categories and Subject Descriptors

K.5.1 [**LEGAL ASPECTS OF COMPUTING**]: Hardware / Software Protection—*Licensing*
; K.6.3 [**MANAGEMENT OF COMPUTING AND INFORMATION SYSTEMS**]: Software Management—*Software Selection*

General Terms

Legal Aspects, Experimentation

Keywords

Software License, Open Source Software, Reuse, Copy and Paste

1. INTRODUCTION

Source code of open source software (i.e. OSS) is available for anyone to modify or redistribute. According to the growth of OSS development [16], software developers can reuse huge amount of OSS source code nowadays.

Everyone has to adhere to the license of a software product when he/she obtains or uses the product. The license of a product expresses the intent of the right holder; Several OSS licenses require the derivative works to apply the same license of the original product, i.e. copyleft[1]. Since different right holders have different intents, many OSS licenses are used.

Software reuse is recognized as a practice for reducing the development cost and improving the product quality. Software reuse happens at several levels of granularity; from simple copy and paste of code snippet, to whole the inclusion, to subsystem reuse.

Software reuse must adhere the license of the reused product. Furthermore, developers must pay attention not to violate the licenses of the product they are developing by reusing other software.

OSS licenses have different attitudes toward reuse.[14] For example, the GNU General Public Licenses (GPL)[5] requires derivative works, including a product that contains a code fragment copied from a GPL product, to be distributed under the GPL. On the other hand, the BSD license[13] only requires that the copyright notice, the license text and the disclaimer are retained. Thus, the reused product must be carefully selected to not conflict with the license of a product under development.

Consequently, software license may affect the frequency of code reuse or the variety of the derived products. For instance, the source code distributed under a copyleft license may be reused less frequently and reused in narrow a variety of software products, compared to non-copyleft license

[1] `http://www.gnu.org/copyleft/` (accessed Oct 2010)

source code. To the knowledge of the available, there is no quantitative study of CnP reuses from the point of view of software license.

This paper presents a preliminary study on the impact of the software licenses on CnP reuse on a small data set. The preliminary experiment intends to assess the process of evaluating CnP source code based on the software license. The target of the experiment is Java source code distributed in Debian GNU/Linux lenny[3]. We investigated copy-and-pasted code fragments that are distributed under three major licenses; the 3-clause BSD license (BSD3), the GPL Version 2 or later (GPLv2+) and the Apache License 2.0 (Apachev2)[1]. To detect copy-and-pasted code fragments, CCFinder[10] and LNR filtering criterion are used.

Result of the experiment shows that source code distributed under the BSD3 or the Apachev2 is more frequently reused than the GPLv2 code. On the other hand, Apachev2 and GPL code has a trend to be reused in code distributed under specific licenses. Through the preliminary experiment, we confirmed that the process of evaluation is effective and applicable for large data set.

The rest of this paper is organized as follows. Section 2 explains background of this study. Section 3 illustrates design and result of the experiment, and Section 4 interprets the result of the experiment. Section 5 discusses about the validity of the experiment, finally, Section 6 shows the conclusion and future remarks.

2. SOFTWARE LICENSE AND REUSING

Nowadays, many open-source licenses are used; for instance, the Open Source Initiative officially approves 67 licenses [2]. This section illustrates three of most major licenses, 3-clause BSD license (BSD3), Apache License 2.0 (Apachev2) and GNU General Public License version 2 (GPLv2) from the perspective of a developer who makes a derivation product. From the point of view of a developer, different licenses mean different restrictions.

When a developer makes a derivative work from a BSD3 product, the former should retain the copyright notice and the full text of the license.

If a developer makes a derivative work from an Apachev2 product, all copyrights, patents, trademarks, and attribution notices should be retained in the new product. Moreover, changed file also should have notices of the modification.

When a developer makes a derivation from a GPLv2 product, the whole derivation must be distributed under the GPLv2 and, changed file should have notices of the modification.

2.1 OSS License and Reuse

Software reuse is recognized as a practical method for developing high quality software with low cost. A developer can reuse source code of OSS products since the source code is easily available.

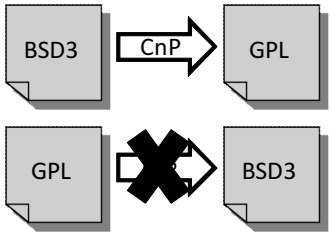

Figure 1: Reusing source code in a different license product

Copy-and-paste (CnP) is one of the most frequently performed methods of reuse. In CnP reuse, source code is copied, modified if needed, and finally, used as a part of new product.[11, 14]

When reusing existing software, both the license of the product being reused and of the developing product must be satisfied. In case the two licenses are incompatible, both cannot be satisfied simultaneously. For example, Apachev2 products cannot be incorporated into the GPLv2 products, since several requirements in the Apachev2 conflict with a clause in GPLv2.[3]

Furthermore, even if the licenses do not conflict, an OSS product cannot be reused if license of the developing product cannot be changed. For example, GPLv2 source code cannot be incorporated into a BSD3 product. In contrast, BSD3 source code can be incorporated into a GPLv2 product because the restrictions of the BSD3 are included in the GPLv2. (Figure 1)

2.2 Impact of Software License on Copy and Paste Reuse

As described above, the license of the reused product is the main concern to software reuse. If a license of source code doesn't match a developing product, the source code cannot be reused in the developing product unless the license of the developing product is changed. Therefore, it is clear that the reusability of software depends on not only functionality or quality, but also license of the software.

Whether reusing source code by CnP is allowed or not depends on its license. As previously explained, when a developer reuses source code, the developer must observe the source codes license. Licenses with very restrictive terms (i.e. GPL) might make it difficult to satisfy their condition, while licenses with more relaxed conditions have easier to meet requirements.

We make the following assumptions:

- source code with a relaxed license is reused under various licenses.

[2] `http://www.opensource.org/licenses` (accessed Oct 2010)

[3] `http://www.gnu.org/licenses/license-list.html` (accessed Oct 2010)

Table 1: kept packages and excluded packages

kept packages	excluded packages
antlr3	antlr
asm3	asm, asm2
db4.6	db4.2, db4.3
junit4	junit
tomcat6	tomcat5.5

- source code with a relaxed license is reused more frequently than source code under a strict license.

3. EXPERIMENT

The goal of this study is an investigation of the impact of licenses on CnP reuse in OSS. To achieve the goal, we need to analyze actual OSS to identify the relationship between licenses and CnP reuse. In this paper, we analyzed the source files of OSS in order to validate our method.

This study focused only source code reuse across applications. We believe that the impact of license on source code reuse within an application is small. Because, even though there are CnP between files under different license, a license problem occurred by these files must be resolved.

3.1 Analysed Code

We selected a part of the source files of Debian/GNU Linux as our analysis target for the following reasons:

- Various licenses are used by it.
- It includes different type of software.

First, we downloaded the packages contained in the main section of Debian/GNU Linux; second, we extracted the content of the ".tar.gz" files; finally, we selected the source code written in Java(.java) the target of our analysis. Consequently, the analysis target consisted of 77452 files (8530896 LOC) from 452 packages.

Note that we kept one of several packages having different versions respectively and excluded the others in the analysis target. For example, between asm, asm2 and asm3, we kept asm3 and excluded asm and asm2 from the analysis target. Table 1 shows the kept packages and the excluded packages.

3.2 Experiment Method

Figure 2 shows an overview of the method of this experiment.

Step 1 We identified the licenses of each file by analyzing the description specifying its licenses in the file. We used Ninka[7] to identify the licenses. Ninka is a tool which analyzes descriptions specifying licenses in a source file and identifies specified licenses. The reason why Ninka was used in this step is that Ninka is a state-of-art license identification tool and more precise than other existing license identification tools such as FOSSology[8]. Note that not only a source file but also a package can have a license. According to the Debian Policy[4], Each package has a 'copyright' file specifying the license of the package. However, this step didn't use this 'copyright' file because the license of source files in the package does not always correspond to the license of the package[6, 8].

Step 2 We extracted the clone sets[10] created by copy-and-paste. A clone set is an equivalence class of the clone-relation. The clone-relation is defined as an equivalence relation. The clone-relation holds between two code portions if (and only if) they are the same sequence. A code fragment which is similar to another is called code-clone. We give a detailed description of the method for extracting clone sets in Section 3.3.

Step 3 We extracted clone sets including a code fragment under specific license A, and we classified and counted the code fragments in these clone set based on their license. Since the source of the CnP cannot be retrieved from code clones, the direction of CnP was not considered in this experiment.

Table 2: Distribution of licenses in all files of the analysis target

License Name	#File	
Apachev2	16350	✓
GPLv2+	8160	✓
LesserGPLv2.1+	6534	
GPLnoVersion,GPLv2+,LinkException	5887	
GPLv2	3222	
BSD3	2181	✓
GPLv2,ClassPathException	1498	
No description specifying license in the file	15813	
Fail to analyze the description	6862	
SeeFile	2786	

Due to time limitation, we investigated only some licenses that are widely used and differ from each other in the condition on CnP reuse. Table 2 shows the abbreviated names of the licenses ranked in descending order by the number of files under it in the analysis target. Licenses marked with a check are the ones we investigated. Table 3 shows the full names of the licenses in Table 2. When we show a version of license, "v<number>" follows. In addition, if users can choose this version or any later, "+" follows. Furthermore, if user can choose several licenses, usable license names follow after ",".

We analyzed Apachev2, GPLv2+ and BSD3 as Table 2 shows. Apachev2 is used by the largest number of files in the analysis target; GPLv2+ and GPL derivatives are counted in the second. BSD3 is used by the largest number of files except files under Apachev2 or licenses which conditions like GPLv2+ such as "LesserGPLv2.1+", "GPLnoVersion, GPLv2+, LinkException", "GPLv2+".

3.3 CnP Detection

We used CCFinder[10] for detecting code fragments created by CnP. CCFinder is a code-clone detection tool. We can

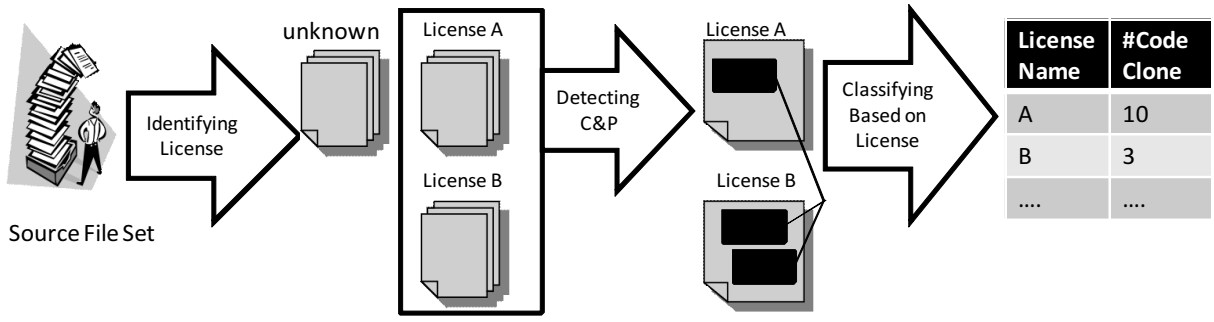

Figure 2: Overview of evaluation method

Table 3: License name abbreviations

Abbreviation	Name
Apache	Apache Public License
BSD3	Original BSD minus advertisement clause
ClassPathException	GNU Classpath License
CPL	Common Public License
GPL	General Public License
LesserGPL	Lesser General Public License
LibraryGPL	Library General Public License
LinkException	GPL linking exception
MITX11noNotice	MIT License/X11License
MPL	Mozilla Public License
MX4jLicense	MX4J License
publicDomain	Public Domain
SeeFile	File Points to another where the its license is
subversion	Subversion License

extract code fragments copied and pasted by detecting code-clones.

In this experiment, when we detect CnP from the analysis target, we detected code-clone ignoring the identifier names, because we wanted to detect code fragments in which identifier names changed after the CnP.

Additionally, we used LNR to filter clone set involving code fragment created by CnP from extracted clone set. LNR is the number of tokens of non repeated elements in a code fragment. A code fragment which LNR is small might includes only variable declarations, assignments or getter/setter declarations. These code fragments are called language specific clones. By contrast, if the LNR of a code fragment is large, the code fragment has higher classes to be created by CnP. Therefore, in this experiment, we presumed that a clone set is created by CnP if the average of LNR of code fragments is over some specific value. This value was set at 50, because our experience shown that 50 is an appropriate value to exclude language specific clones.

3.4 Results

Table 4 shows the number of code fragments under each license; Table 5 shows the case of Apachev2; Table 6 shows the case of GPLv2+.

Table 7 shows the number of code fragments, files and the ratio of code fragments to files of BSD3, Apachev2 and GPLv2+. We can see from Table 7 that the order arranged in descending order of number of code fragments compared to number of files is BSD3, Apachev2, GPLv2+.

Table 4: Distribution of code fragments having clone-relation to files under BSD3

License Name	#Fragments
BSD3	613
GPLv2+	20
Apachev2	16
LibraryGPLv2+	14
GPLv2,ClassPathException	1
LesserGPLv2.1+	1

Table 5: Distribution of code fragments having clone-relation to files under Apachev2

License Name	#Fragments
Apachev2	1533
Apachev1.1	316
LesserGPLv2.1+	42
MPLv1_1	33
BSD3	29
MX4JLicensev1	16
GPLv2+	4
LibraryGPLv2+	3
MPLv1_0	2
MITX11noNotice	2
publicDomain	1
subversion+	1
EPLv1	1

We found that CnP fragments tend to have the same license. In the case of BSD3, code fragments under BSD3 account for 92% of all code fragments. Similarly, in the case of Apachev2, code fragments under Apachev2 account for 77% of all. In the case of GPLv2+, code fragments under GPLv2+ account for 48% of all.

In the case of Apachev2, code fragments under Apachev1.1 account for 16% of all. Similarly, in the result of GPLv2+, "GPLnoVersion, GPLv2+, LinkException" account for 41% of all.

If we evaluate the results from the point of view of the number of licenses, Apachev2 has CnP relationship to the largest number of licenses. Apachev2 has CnP relationship to 13 licenses. BSD3 and also GPLv2+ have CnP relationship to 6 licenses.

Table 6: Distribution of code fragments having clone-relation to files under GPLv2+

License Name	#Fragments
GPLv2+	268
GPLnoVersion,GPLv2+,LinkException	225
BSD3	28
LibraryGPLv2+	20
Apachev2	4
LesserGPLv2.1+	4

Table 7: Number of code fragments and files

	#Fragments	#File	#Fragments/#File
BSD3	665	2181	0.304906
Apachev2	1983	16350	0.121284
GPLv2+	549	8160	0.067279

4. DISCUSSION

In the result of Apachev2, there is large number of code fragments having CnP relationship to code fragments under Apachev1.1. We believe that Apachev1.1 has been changing to Apachev2 currently because Apachev1.1 is an old version of Apachev2.

GPLv2+ has the smallest number of #Fragments/#File in table 7. This result shows that GPLv2+ is reused less frequently than BSD3 and Apachev2. We believe that code fragments under other licenses are copied into code under GPLv2+, however, there is little case that a code fragment under GPLv2+ is copied into code under another license not in the GPL family. We believe the reasons are:

- Copy-and-Pasting a code fragment under GPLv2+ to code under another license except GPLv2+ and GPLv3 violates the condition of GPLv2+.
- Copy-and-Pasting a code fragment under Apachev2 or BSD3 in code under GPLv2+ is permitted.
- Copy-and-Pasting a code fragment under GPL family such as LesserGPL or "GPLnoVersion, GPLv2+, LinkException" and changing their license to GPL is permitted.

Code fragments under BSD3 or Apachev2 are reused more frequently than code fragments under GPLv2+. We believe that it is easier to satisfy the conditions of the BSD3 or the Apachev2 than these of the GPLv2+. In fact, code fragments under the Apachev2 have CnP relationship to code fragments under more licenses than code fragments under the GPLv2+. Therefore, we suggest that code fragments under Apachev2 are copy-and-pasted frequently into code under another license.

In section 3.4, we shown that all licenses share the common characteristic that code fragments under each of the own license have the highest proportion in each result of investigations. We suggest that there are many cases that code fragments are copy-and-pasted to code fragments created by a developer in same development organization.

There are many open source licenses as described in Section 2, the developers of the OSS have to select a license from the many available. The result of the investigation may be contributory to license selection.

This experiment is a preliminary study. Therefore, we plan to perform an experiment on a larger analysis target. It is possible to detect CnP in larger source code because we can split a set of source code to decrease their size as possible as to analyze by CCFinder, and merge these results. On the other hand, identifying license of code fragment in large object is also possible. Because Ninka analyzes files one by one.

5. THREAT TO VALIDITY

It is possible that the result of the experiment depends on the CnP detection capability of code clone detection tool CCFinder[10]. We are motivated to use CCFinder in the experiment because one of main application of code clone detection tool is CnP detection, and the usefulness of CCFinder is shown in Bellon's benchmark[2]. As future work, comparative experiments with other available code clone detection tools (e.g., CP-Miner[12]) should be performed.

The result of the experiment also depends on the characteristic of metric LNR and its threshold value. We use the metric LNR to exclude language-dependent clones (e.g., repeated setter/getter invocations in Java programs). As future work, we should perform the experiments to show the effect of threshold values of metric LNR. RNR[9] is another metric to exclude language-dependent clones. Currently, we are planning to perform the comparative experiment with LNR and RNR.

Ninka is employed to identify the licenses of source files. We believe that using Ninka is valid because the accuracy of Ninka is good; In [7], recall was 82% and the precision was 96%.

However, Ninka cannot detect a license if a source file contains no description about license. In addition, Ninka also cannot detect a license if the license is not registered in the database. In this experiment, source files that Ninka couldn't detect their license are removed from the target of detecting CnP. Hence, source files which were not detected their license didn't influence results.

The analysis target is a small portion of source files in Debian GNU/Linux. Therefore, this result may include a lot of sampling error. For this reason, we believe that applying this result to general OSS is not acceptable.

6. CONCLUSION AND FUTURE WORK

This paper performed a preliminary study on impact of software licenses on CnP reuse. Open-source Java source code in Debian/GNU Linux was analyzed and classified based on their license. In particular, copied code fragments that relate to the source files distributed under 3-clause BSD license, Apache License version 2.0 or GPL version2 or lator were analyzed.

The result of the experiment shows that most of the code

fragments are copied to files distributed under same license or relative licenses that is designed by same organization. On the other hand, comparing to BSD3 and GPLv2+, Apachev2 code fragments are copied into files that are distributed under a various licenses. By contrast, almost all GPLv2+ code are copied between the files distributed under the GPL family licenses.

We are planning a large scale experiment. Especially, the all source code and licenses in OSS products in Debian GNU/Linux is a target of the experiment.

This paper focused on clone sets created by CnP. Thus, the direction of copying is not identified. To clarify the true impact of the license to copy-and-pasting, origin analysis is required to know the direction.

To identify the developing organization or the developer who copied a fragment is also important to clarify the impact.

In this experiment, we didn't discriminate between code clone generated by copy-and-pasted and code clone generated by including library in source form. Discriminating these code clone, it will be clear that difference of an impact of license on their cases. Therefore, we plan to retrieve code clone generated by including library's source code using with FCFinder[15].

7. ACKNOWLEDGMENTS

This work has been supported by Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Young Scientists (B) (No.21700031), and Grant-in-Aid for Research Activity start-up (No.22800040).

8. REFERENCES

[1] Apache Software Foundation. Apache license, version 2.0. `http://www.apache.org/licenses/LICENSE-2.0` Accessed Oct 2010.

[2] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo. Comparison and evaluation of clone detection tools. *IEEE Trans. Softw. Eng.*, 33(9):577 –591, Sept. 2007.

[3] Debian Project. Debian gnu/linux. `http://www.debian.org/` Accessed Oct 2010.

[4] Debian Project. Debian policy manual. `http://www.debian.org/doc/debian-policy/` Accessed Oct 2010.

[5] Free Software Foundation. GNU general public license. `http://www.gnu.org/licenses/gpl.html` Accessed Oct 2010.

[6] D. M. German, M. Di Penta, and J. Davies. Understanding and auditing the licensing of open source software distributions. ICPC '10, pages 84–93, 2010.

[7] D. M. German, Y. Manabe, and K. Inoue. A sentence-matching method for automatic license identification of source code files. In *ASE 2010*, pages 437–446, 2010.

[8] R. Gobeille. The fossology project. In *Proceedings of the 2008 international working conference on Mining software repositories*, MSR '08, pages 47–50, New York, NY, USA, 2008. ACM.

[9] Y. Higo, T. Kamiya, S. Kusumoto, and K. Inoue. Method and implementation for investigating code clones in a software system. *Information & Software Technology*, 49(9-10):985–998, 2007.

[10] T. Kamiya, S. Kusumoto, and K. Inoue. CCFinder: A multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions on Software Engineering*, 28:654–670, 2002.

[11] J. Li, R. Conradi, C. Bunse, M. Torchiano, O. P. N. Slyngstad, and M. Morisio. Development with off-the-shelf components: 10 facts. *IEEE Software*, 26:80–87, 2009.

[12] Z. Li, S. Lu, S. Myagmar, and Y. Zhou. Cp-miner: finding copy-paste and related bugs in large-scale software code. *IEEE Trans. Soft. Eng.*, 32(3):176 – 192, March 2006.

[13] Open Source Initiative. The BSD license. `http://www.opensource.org/licenses/bsd-license.php` Accessed Oct 2010.

[14] M. Ruffin and C. Ebert. Using open source software in product development: A primer. *IEEE Software*, 21(1):82–86, 2004.

[15] Y. Sasaki, T. Yamamoto, Y. Hayase, and K. Inoue. Finding file clones in freebsd ports collection. In *Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on*, pages 102 –105, 2010.

[16] W. Scacchi. Free/open source software development: recent research results and emerging opportunities. In *ESEC/FSE 2007*, pages 459–468, 2007.