A Study on the Safe Architecture Design Strategies of the Proprietary Software through Analysis of GPL License Family

Jun Seok Park¹, Soo Hong Kim²

¹ Dept. Of Computer Information & Communication Engineering, Sangmyung University.
20, Hongjimun 2-gil, Jongno-gu, Seoul, Korea
Jun Seok Park, dkaien77@hanmail.net
² Dept. Of Computer Software Engineering, Sangmyung University.
31, Sangmyungdae-gil, Dongnam-gu, Cheonan, Chungnam, Korea
Soo Hong Kim, soohkim@smu.ac.kr

Abstract. Open Source Software (OSS) is being used extensively around the world with the emergence of its various advantages, but the researches on the violation of OSS licenses are focused on the areas of law, patent and economy, and it is rare to find a research paper on the violation of structural licenses that can occur in the actual development of software. We have analyzed cases of violation of licenses in GPL group, the typical OSS licenses, schematized it to develop a model based on which will be used to study architecture design strategies to avoid violation of copyright.

Keywords: open source software, proprietary software, license conflict, license violation, snippet, license infection

1 Introduction

Open Source Software(OSS) is freely available to anyone free of charge and is a program whose source codes are disclosed. However it does not necessarily mean that borrowing the original copyright holder’s source codes and reproducing them will be freely permitted. OSS’s source codes are disclosed, but as shown in Fig. 1, they are works under the protection of intellectual property law such as copyright[1]. Nevertheless, many developers or users misunderstand and think they can use OSS licenses loosely since the source codes are disclosed free of charge providing little protection again infringement of copyright. Particularly, such violations are monitored world-wide by OSS organizations such as SFLC(software freedom law center)[2] resulting in lawsuits for violations of various licenses. Domestic companies such as Samsung Electronics and Humax was sued by SFLC in New York District Court[3].

The researches in the field so far focused on stages prior to the development or after the distribution resulting in issues such as methods to classify licenses, considerations to apply licenses, the relationship between licenses and patents and legal solutions in cases of clashes between licenses rather than the point of development.
The research started with this point, analyzed violations of GPL-group licenses, the typical OSS licenses, applied the characteristics to build a schematized model, defined cases where licenses violation can arise or not and proposed appropriate design strategies based on them.

Chapter 2 reviews the flow of preceding researches on clashes between OSS licenses that were conducted so far, and chapter 3 analyzes requirements for GPL licenses. Chapter 4 schematizes models for each license using the analysis results of chapter 3 and proposes safe architecture design strategies of software applying them to arrive at conclusions.

Fig. 1. Classification of Software

2 Preceding Research

The existing researches on OSS mainly measured the economic impact of OSS or dealt with application methods of OSS. In particular, in the beginning of 2000s, a number of researches were conducted on the introduction of projects using OSS. Coughlan studied development methods of sustainable community composition and cooperation in terms of business model from a perspective of project management methods[4]. However, this research applied general project management method to the OSS environment not escaping from researches on software development methods. Full-scale researches on the clashes between OSS licenses were conducted by Gordon who studied compatibility of OSS licenses. The research analyzed characteristics of difference licenses for each module existing within software module developed internally and covered issues of license compatibility and clash issues[5]. However, this research was limited to licenses regarding borrowed module falling short of analyzing characteristics of other projects. Technical researches of module units include a research of Mathur analyzing 1,423 projects for the possibility of clashes between source codes suggesting that GPL-group licenses occupy a high percentage of overall projects and fairly in accordance in a relative sense[6]. Nevertheless, the compliance of GPL licenses is too extensive to be decided by certain investigations for its rate of compliance lacking legitimate investigative methods. Also, the research studied possibility of clashes based on text matching in unit files or mathematical inferences failing to propose specific strategies to prevent clashes between licenses.
3 Analysis of the GPL License Family

When using OSS for proprietary software, a deep understanding of license requirements is a must. This chapter examines the characteristics of GPL-group licenses, the typical OSS licenses and utilized them as basic data to develop a model of disclosure scope of source codes.

GPL (General Public License) 2.0 is selected by the highest number of OSS presently, issued by FSF (Free Software Foundation) [7], and awarded liberties to freely use, duplicate, distribute and modify. However, they must disclose all codes related to GPL codes, thus carrying more strict requirements than other licenses.

GPL (General Public License) 3.0 succeeded GPL 2.0 in essence, but added features related to DRM, software patents and compatibility for more flexible utilization. Notably, it was modified to be compatible with Apache License 2.0 that is used in various fields such as cloud computing, big data and Android.

LGPL (Lessor General Public License) is a strategically developed license to encourage OSS use through free use of OSS library. Basically, all codes related to LGPL codes must be disclosed as a LGPL license, but are not required to do so when they are linked by a library.

4 Architecture Design of the Proprietary Software using OSS

4.1 Model of disclosure scope of GPL and LGPL source codes

In the use of GPL source codes when developing proprietary software, source codes should be disclosed for all the program operated by a process in cases of modification of source codes or link of a new software. This involves inclusion to identical execution files and execution by a link to the shared address domain. LGPL is the same as GPL except that there is no obligation to disclose source codes when codes are linked. However, GPL loses a requirement to disclose source codes when two programs are communicated in the form of independent processes such as pipes, sockets and command-line arguments regardless of link. Certainly, deciding on the requirement to disclose cannot be made by any other than judges, but generally these are the exceptions that apply[8]. Fig. 2 is a model of disclosure scope of source codes schematizing requirements of GPL-group licenses covered in Chapter 3.

![Fig. 2. Model of disclosure scope of GPL and LGPL source codes](image-url)
4.2 Architecture Design strategies of the Proprietary Software using OSS

Based on the above model, codes that can be disclosed in proprietary software include internally developed codes in the main program. In cases of using Linux system call, class path exceptions or loadable device drivers written in kernel module, there is no obligation to disclose. Additionally, when processes are written in independent forms using pipes, sockets and command-line argument methods regardless of link, the obligation to disclose sources codes expires even if they are linked to GPL.

Naturally, thorough consideration of content and meaning of communication will reveal that exchanging intricate internal data structures can be regarded as a large one program with two parts possibly producing exceptions. Nonetheless, most of the cases do not fall into the category of exceptions. Fig. 2 is a model of disclosure scope in the case of link to GPL license.

To apply LGPL to proprietary software, codes that will be disclosed can be included in the internally developed codes in the main program while source codes that will not be disclosed can be maintained as link. However, with regard to LGPL, static link should provide object codes of application programs for user to modify library and run identical execution files. In other words, for the static link of sub program 1, object codes of sub program 1 should be disclosed.

5 Conclusion

This research created a model schematizing requirements of GPL-group licenses to propose architecture design strategies for proprietary software utilizing the model. Additional analysis of MPL, EPL, BSD and Apache License, not only GPL and LGPL, and the analysis of actual cases of violations through real software verification will lead to further researches on the architecture design strategies for general proprietary software utilizing OSS.

References