2015 IEEE/ACM 37th IEEE International Conference on Software Engineering

A Large Scale Study of License Usage on GitHub

Christopher Vendome
The College of William and Mary,
Williamsburg, VA, 23185
cvendome@cs.wm.edu

***Abstract*—The open source community relies upon licensing in order to govern the distribution, modification, and reuse of existing code. These licenses evolve to better suit the requirements of the development communities and to cope with unaddressed or new legal issues. In this paper, we report the results of a large empirical study conducted over the change history of 16,221 open source Java projects mined from GitHub. Our study investigates how licensing usage and adoption changes over a period of ten years. We consider both the distribution of license usage within projects of a rapidly growing forge and the extent that new versions of licenses are introduced in these projects.**

***Index Terms*—Software Licenses, Mining Software Repositories, Empirical Studies**

I. Motivation and Problem

Utilizing existing open source code can allow developers to integrate stable features that other developers have written within their own systems. However, the developer looking to reuse, distribute, or modify the existing code must comply with all licensing restrictions, including those of all dependencies. These *licensing statements* are either appended to the beginning of a file as a comment or they are in a separate file, such as "COPYING" or "LICENSE." The problem arises when developers use two licenses that are incompatible.

A canonical example of license incompatibility is between the GPL_{v2} and the $Apache_{v2}$ license, due to patent clauses that protect the copyright holder in the latter license. To further complicate matters, new versions of these licenses can be released and break compatibility. For example, the $LGPL_{v3}$ is not compatible with the GPL_{v2}, unless the original author adds an exception to allow the usage of future license version (we classify these licenses as GPL_{v2+}) [5]. Additionally, the original authors are not forced to the accepted set of open source licenses, but they can write their own terms.

In our study, we investigate the relative license usage over a ten year period in which several new license versions were released for 16,221 projects on GitHub [10]. We investigate the proportions of different kinds of licenses that are introduced by open source projects hosted on GitHub. Our goal is to observe potential trends in license usage (adoption and survival). For example, we are interested in the impact of the release of a new license version and we present possible reasons for either their rapid usage or the lack of adoption and survival of earlier versions.

II. Related Work

In terms of license identification and classification, Di Penta *et al.* [3] utilized code search to infer the license of byte-code of jars and combined it with textual analysis to automatically identify the jar's license. The FOSSology project [11] created a license identification tool FOSSlic using an aliment matrix to identify licenses from textual files. Tuunanen *et al.* [16] developed the tool, ASLA, with 89% accuracy in determining license declarations in order to extract the licenses from open source projects. German *et al.* [9] presented a pattern-matching tool, Ninka, which achieved a precision of 96% at detecting licenses from text files.

In terms of studies on license adoptions and evolution, Di Penta *et al.* [4] investigated the way in which licenses migrate during maintenance and evolution of a system. The authors were unable to determine a generalized pattern, but they identified changes to both license type and license version as the systems evolved. German *et al.* [8] investigated how developers handle incompatible licenses and created a model. The model aimed to categorize the applicability and the strength and weakness license patterns. German *et al.* [6] investigated the packages of the Fedora-12 Linux distribution. Their studied sought to determine if declared package licensing and source code licensing was consistent for each package, and they were able to verify a subset of license incompatibilities due to dependencies by contacting the Fedora developers.

German *et al.* [7] investigated if cloned code fragments were being properly cloned in accordance with the licensing terms between either OpenBSD or FreeBSD and the Linux Kernel. The authors observed that code migrations were valid since the code was changing from a more permissive license to a more restrictive license, which did not cause licensing conflicts. Finally, Manabe *et al.* [12] observed changes in licenses for FreeBSD, OpenBSD, Eclipse, and ArgoUML, but they found that the patterns were system-specific and did not generalize across the systems.

III. Approach and Uniqueness

The approach is unique in the way that we classify a license adoption within a project and analyze license survival. Instead of considering project-level licenses, we are interested in the first occurrence of a particular license being attributed to a file within a system. The approach has three main steps: 1) project cloning, 2) license extraction, and 3) license analysis. We automatically mined the metadata of all public projects on GitHub and identified 381,161 projects as Java projects. From this list of projects, we randomly cloned 16,221 unique Git repositories (441 Gigabytes of storage space) for our dataset. Projects were not a fork of another project within the dataset.

978-1-4799-1934-5/15 $31.00 © 2015 IEEE
DOI 10.1109/ICSE.2015.245

ICSE 2015, Florence, Italy
ACM Student Research Competition

Subsequently, we utilized the MARKOS [2] code analyzer to extract the licenses of each files at commit-level granularity. The code analyzer relies on Ninka [9], which is that current state-of-the-art tool for license identification. Each commit was linked to its timestamp so that we could identify the year that each license was introduced into at least one file of a particular project. Finally, we analyzed the data from each project to identify all of the licenses of a project and when each license was first committed. We consider the first occurrence instead of number of files with a license to remove the bias that larger systems would assert on our analysis.

In our analysis, we distinguish each license version as different licenses. For example, if a project was created with only files licensed under the GPL_{v2} in 2010 and during 2012 a file licensed under the GPL_{v3} was introduced, we would consider these as two distinct events. The former would be captured within the number of GPL_{v2} projects for the year 2010, while the latter would be captured within the number of GPL_{v3} projects for the year 2012. For CMU and BSD licenses, Ninka only can detect that the license is a variant of each of these licenses so we only report the first adoption of the license family as opposed to family and version. Our approach considers file-level analysis, since a previous study about GitHub indicated that only 14.9% of projects have a license file [13]. Therefore, it is not reliable to only investigate files like `License.txt`.

IV. Results

Fig. 1 depicts the *relative usage* of licenses per year between 2002 and 2012 at project-level granularity (i.e. each license can only be counted at most once per project). We exclude projects prior to 2002 since our dataset had insufficient representation before this time period. Earlier data was manually investigated for reliability. In 2002, $LGPL_{v2.1}$ and $LGPL_{v2.1+}$ (a variant of $LGPL_{v2.1}$ that allows accepting future versions of the LGPL) licenses are the widest adopted licenses. They are followed by MIT/X11, GPL_{v2+}, CPL_{v1}, and $Apache_{v1.1}$. At this time, more permissive license (ex: MIT/X11, Apache, and LGPL) adoption was more prevalent as compared to more restrictive license (ex: GPL) adoption. LGPL allows linking the licensed software to systems that are not not licensed under the LGPL or the GPL as a library (if the original LGPL code is modified, the modified code must be licensed LGPL); whereas, the GPL requires systems using GPL code to any extent to also be licensed under the GPL.

Between 2004 and 2008, several new open source licenses were released (e.g., $Apache_{v2}$, $CDDL_{v1}$, EPL_{v1}, GPL_{v3}, $LGPL_{v3,}$ and $DWTFYW_{v2}$),and the Apache ecosystem started its exponential growth [1]. Consequently, we observe fast growth and diffusion of the $Apache_{v2}$ license, moving from percentages of diffusion near 15% up to 40% of projects in our sample.

In the same time period, despite the introduction of the newer version, $v3$ of the GPL license, a high percentage or projects still adopted the old $GPL_{v2(+)}$. To a larger extent, we observed the same phenomenon for the LGPL, for which $v2$

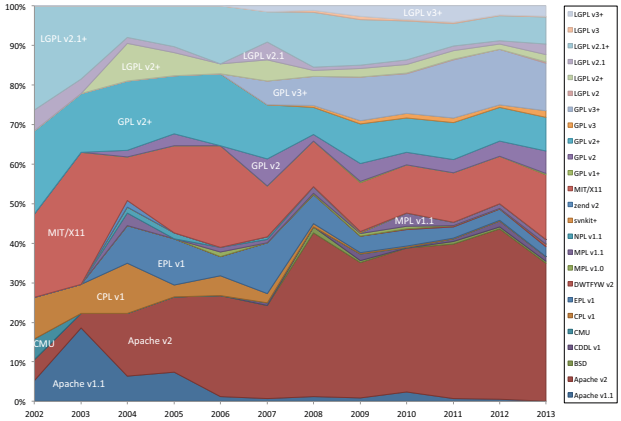

Fig. 1. Relative License Usage between 2002 and 2012.

and $v2.1$ dominate compared to the newer $v3$. The $LGPL_{v3}$ inherits from GPL_{v3}, providing additional list of exceptions. Because of this interconnectivity, it is more difficult for developers to understand the implications as compared to the $LGPL_{v2}$, which utilizes a less litigious language. This interpretation of the two licenses and the difficulty is corroborated by discussions among developers that we found online [14].

Similarly to the $LGPL_{v3}$, the GPL_{v3} introduces several new clauses, which make the license far more complicated [15] in its understanding as compared with the GPL_{v2}. Additionally, the GPL_{v3}'s block of *tivoization* (i.e. utilizing hardware that can detect modifications to code to prevent the system from operating) may have contributed to the survival of the GPL_{v2}. Finally, the GPL_{v3} is not compatible with the GPL_{v2} so existing systems licensed under the GPL_{v2} cannot incorporate GPL_{v3} without migrating licenses.

Additionally, we observe a nearly complete disappearance of CPLv1 in favor of its new version, EPLv1, during this time period. The difference between these licenses primarily represents a textual replacement of the term "Common" by "Eclipse" and "IBM" by "the Eclipse Foundation." Noticeably, this confirms previous finding Di Penta *et al.* [4] limited to the Eclipse-JDT project. Finally, the prevalence of the MIT/X11 license decreases during the explosion of the $Apache_{v2}$ license. This observation potentially demonstrates the open source community's need to have a license that offers more legal protection, while still being permissive. For example, the $Apache_{v2}$ license grants the original author patent protection.

V. Contributions

We empirically studied phenomena related to license usage across a set of 16,221 Java projects hosted on GitHub. We demonstrated that new license versions were quickly adopted by developers whenever licenses did not introduce strong changes/restrictions with respect to the previous versions (e.g., $Apache_{v1.1}$ vs $Apache_{v2}$). On the contrary, new license versions introducing new clauses and constraints with respect to their predecessors (e.g., GPL_{v3} vs GPL_{v2}) favored the long survival of earlier versions.

REFERENCES

[1] G. Bavota, G. Canfora, M. Di Penta, R. Oliveto, and S. Panichella. The evolution of project inter-dependencies in a software ecosystem: The case of apache. pages 280–289, 2013.

[2] G. Bavota, A. Ciemniewska, I. Chulani, A. De Nigro, M. Di Penta, D. Galletti, R. Galoppini, T. F. Gordon, P. Kedziora, I. Lener, F. Torelli, R. Pratola, J. Pukacki, Y. Rebahi, and S. G. Villalonga. The market for open source: An intelligent virtual open source marketplace. In *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering, CSMR-WCRE 2014, Antwerp, Belgium, February 3-6, 2014*, pages 399–402, 2014.

[3] M. Di Penta, D. M. Germán, and G. Antoniol. Identifying licensing of jar archives using a code-search approach. In *Proceedings of the 7th International Working Conference on Mining Software Repositories, MSR 2010 (Co-located with ICSE), Cape Town, South Africa, May 2-3, 2010, Proceedings*, pages 151–160, 2010.

[4] M. Di Penta, D. M. Germán, Y. Guéhéneuc, and G. Antoniol. An exploratory study of the evolution of software licensing. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, ICSE 2010, Cape Town, South Africa, 1-8 May 2010*, pages 145–154, 2010.

[5] Free Software Foundation. Categories of free and nonfree software. https://www.gnu.org/licenses/license-list.html.

[6] D. M. Germán, M. Di Penta, and J. Davies. Understanding and auditing the licensing of open source software distributions. In *The 18th IEEE International Conference on Program Comprehension, ICPC 2010, Braga, Minho, Portugal, June 30-July 2, 2010*, pages 84–93, 2010.

[7] D. M. Germán, M. Di Penta, Y. Guéhéneuc, and G. Antoniol. Code siblings: Technical and legal implications of copying code between applications. In *Proceedings of the 6th International Working Conference on Mining Software Repositories, MSR 2009 (Co-located with ICSE), Vancouver, BC, Canada, May 16-17, 2009, Proceedings*, pages 81–90, 2009.

[8] D. M. Germán and A. E. Hassan. License integration patterns: Addressing license mismatches in component-based development. In *31st International Conference on Software Engineering, ICSE 2009, May 16-24, 2009, Vancouver, Canada, Proceedings*, pages 188–198, 2009.

[9] D. M. Germán, Y. Manabe, and K. Inoue. A sentence-matching method for automatic license identification of source code files. In *ASE 2010, 25th IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, September 20-24, 2010*, pages 437–446, 2010.

[10] Github, Inc. https://github.com.

[11] R. Gobeille. The FOSSology project. In *Proceedings of the 2008 International Working Conference on Mining Software Repositories, MSR 2008 (Co-located with ICSE), Leipzig, Germany, May 10-11, 2008, Proceedings*, pages 47–50, 2008.

[12] Y. Manabe, Y. Hayase, and K. Inoue. Evolutional analysis of licenses in FOSS. In *Proceedings of the Joint ERCIM Workshop on Software Evolution (EVOL) and International Workshop on Principles of Software Evolution (IWPSE), Antwerp, Belgium, September 20-21, 2010.*, pages 83–87, 2010.

[13] N. McAllister. Study: Most projects on github not open source licensed http://www.theregister.co.uk/2013/04/18/github_licensing_study.

[14] StackExchange. Lgpl 2.1 vs lgpl 3.0 advantages and disadvantages, http://programmers.stackexchange.com/questions/189668/lgpl-2-1-vs-lgpl-3-0-advantages-and-disadvantages.

[15] StackOverflow. What are the differences between gpl v2 and gpl v3 licenses? [closed], http://stackoverflow.com/questions/41460/what-are-the-differences-between-gpl-v2-and-gpl-v3-licenses.

[16] T. Tuunanen, J. Koskinen, and T. Kärkkäinen. Automated software license analysis. *Autom. Softw. Eng.*, 16(3-4):455–490, 2009.