2015 IEEE 23rd International Conference on Program Comprehension

License Usage and Changes: A Large-Scale Study of Java Projects on GitHub

Christopher Vendome[1], Mario Linares-Vásquez[1], Gabriele Bavota[2],
Massimiliano Di Penta[3], Daniel German[4], Denys Poshyvanyk[1]
[1]The College of William and Mary, VA, USA — [2]Free University of Bolzano, Italy
[3]University of Sannio, Italy — [4]University of Victoria, BC, Canada

***Abstract*—Software licenses determine, from a legal point of view, under which conditions software can be integrated, used, and above all, redistributed. Licenses evolve over time to meet the needs of development communities and to cope with emerging legal issues and new development paradigms. Such evolution of licenses is likely to be accompanied by changes in the way how software uses such licenses, resulting in some licenses being adopted while others are abandoned. This paper reports a large empirical study aimed at *quantitatively* and *qualitatively* investigating when and why developer change software licenses. Specifically, we first identify licenses' changes in 1,731,828 commits, representing the entire history of 16,221 Java projects hosted on GitHub. Then, to understand the rationale of license changes, we perform a qualitative analysis—following a grounded theory approach—of commit notes and issue tracker discussions concerning licensing topics and, whenever possible, try to build traceability links between discussions and changes. Our results point out a lack of traceability of *when* and *why* licensing changes are made. This can be a major concern, because a change in the license of a system can negatively impact those that reuse it.**

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

I. Introduction

The increasing diffusion of Free and Open Source Software (FOSS) projects is a precious resource for developers, who can reuse existing assets, extend/evolve them, and in this way create new work productively and reduce costs. Nevertheless, whoever is interested in integrating FOSS code in their software project (and redistributing resulting source code with the project itself), or modifying existing FOSS projects to create new work—referred to as "derivative work"—must be aware that such activities are regulated by *software licenses* and in particular by certain FOSS licenses. A file is licensed by adding a *licensing statement* as a comment on top of a file, or in a separate text file that indicates the license(s) under which such file is licensed.

Generally speaking, FOSS licenses can be classified into *restrictive* (also referred to as "copyleft" or "reciprocal") and *permissive* licenses. A restrictive license requires developers to use the same license to distribute new software that incorporates software licensed under such restrictive license (i.e. the redistribution of the derivative work must be licensed under the same terms); meanwhile, permissive licenses allow re-distributors to incorporate the reused software under a difference license [30], [17]. The GPL is a classic example of a restrictive license. In Section 5 of the *GPL-3.0*, the license addresses code modification stating that "*You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy*" [4]. The BSD Licenses are examples of permissive licenses. For instance, the BSD 2-Clause has two clauses that detail the use, redistribution, and modification of licensed code: (i) the source must contain the copyright notice and (ii) the binary must produce the copyright notice and contain the disclaimer in documentation [1].

When developers or organizations decide to make a project available as open source, they can license their code under many different existing licenses or specify a new unique license. The choice may be dictated by the set of dependencies that the project has (e.g., libraries) released under various different licenses. For example, if a project has to (statically) link some GPL code, then it must be released under the same GPL version; failing to fulfill such a constraint would create potential legal implications. Also, as shown by Di Penta *et al.* [14], the choice of the licenses in a FOSS project may have massive impact on its success, as well as on projects using it. For example—as it happened for the IPFilter project [5]—a highly restrictive license may prevent others from redistributing the project (in the case of IPFilter, this caused its exclusion from OpenBSD distributions). An opposite case is the one of MySQL connect drivers, originally released under *GPL-2.0*, whose license was modified with an exception [29] to allow the driver's inclusion in other software released under other open source licenses (e.g., the Apache one). In summary, the choice of the license—or even a decision to change an existing license—is a crucial crossroad point in the context of software evolution of every FOSS project.

In order to encourage developers to think about licensing issues early in the development process, some forges (e.g., GitHub) have introduced specific mechanisms such as the possibility of picking the project license at the time the repository is created, and websites (e.g., http://choosealicense.com/) for helping developers to choose a license. In addition, there are numerous research efforts aimed at supporting developers in classifying source code licenses [22], [21] and identifying licensing incompatibilities [18]. Even initiatives such as the Software Package Data Exchange (SPDX) [6] have been aimed at proposing a formal model for licenses. However, despite of the effort put by the FOSS community, researchers, and independent companies, it turns out that developers usually do not have a clear idea yet on the exact consequences of licensing (or not) their code using a specific license, or they are unsure, for example, on how to re-distribute code licensed with a dual license among the other issues.

Paper contributions. This paper reports the results of a large empirical study aimed at quantitatively and qualitatively investigating when and why licenses change in Java projects hosted on GitHub. In particular, we mined the entire change history of 16,221 projects, extracting the license type (e.g., GPL) and version (e.g., $v2$) in each of the 4,665,611 files involved in a total of 1,731,828 commits. Starting from this data, we provide quantitative evidence on (i) the diffusion of licenses in FOSS systems, (ii) the most common license-change patterns, and (iii) the traceability between the license changes to both the commit messages and the issue tracker discussions; additionally, we provide qualitative rationale on the reasons why developers adopt specific license(s), both for initial licensing and for licensing changes.

Previous work explored license incompatibilities [18], license changes [14], license evolution [25] and integration patterns [20]. Building upon previous results, this paper:

1) Constitutes, to the best of the authors' knowledge, the largest study aimed at analyzing the change patterns in licensing of software systems (earlier work was limited to the analysis of up to six projects [25], [14], whereas this work analyzes 16,221 projects).
2) Is the first work aimed at linking licensing changes to their rationale by means of a qualitative analysis of commit notes and issue tracker discussions.

The study results suggest that determining the appropriate license of a software project is far from trivial and that a community can influence developers when picking a license. Although licensing is considered by developers during the initial release of a project, forges and third party tools do not provide enough support to developers with licensing related tasks, e.g., in picking a license, declaring the license of a project, license migration, traceability of changes related to licensing. Also, there is a lack of consistence and standardization in the mechanism that should be used for declaring a license (e.g, putting it in source code heading comments, separate license files, README files, etc.), which fosters confusion among developers attempting to reuse code. Moreover, the legal nature of the licenses exacerbate this problem since the implications and grants or restrictions are not always clear for developers when the license is present.

II. Related Work

Our work is mainly related to (i) techniques and tools for automatically identifying and classifying licenses in software artifacts, and (ii) empirical studies focusing on different aspects of license adoption and evolution.

A. Identifying and Classifying Software Licensing

To the best of our knowledge, the problem of license identification has firstly been tackled in the FOSSology project [22] aimed at building a repository storing FOSS projects and their licensing information and using a machine learning approach to classify licenses. Tuunanen *et al.* [31] present ASLA, a tool aimed at identifying licenses in FOSS systems; the tool has been shown to determine licenses in files with 89% accuracy.

German *et al.* [21] proposed Ninka, a tool that uses a pattern-matching based approach for identifying statements that characterize various licenses. Given any text file as an input, Ninka outputs the license name (e.g., GPL) and its version (e.g., 2.0). In the evaluation reported by the authors, Ninka's achieved a precision around 95% while detecting licenses. Ninka is currently considered the state-of-the-art tool in the automatic identification of software licenses.

While the typical license classification problem arises when source code is available, in some cases, it is not available—i.e., only byte code or binaries are available—and the goal is to identify whether the byte code has been produced from source code under a certain license. To this aim, Di Penta *et al.* [13] combined code search and textual analysis to automatically determine a license under which jar files were released. Their approach automatically infers the license from decompiled code by relying on Google Code Search.

B. Empirical Studies on Licenses Adoption and Evolution

Di Penta *et al.* [14] investigated the migration of licenses over the course of a project's lifetime. The study suggests that licenses changed version and type during software evolution, but there was no generic patterns generalizable to the six analyzed FOSS projects. German *et al.* [20] analyzed 124 open source packages exploited by several applications to understand how developers deal with license incompatibilities. Based on this analysis, they built a model outlining when specific licenses are applicable and what are their advantages and disadvantages. Later, German *et al.* [18] presented an empirical study focused on the binary packages of the Fedora-12 Linux distribution aimed at (i) understanding if licenses declared in the packages were consistent with those present in the source code files and (ii) detecting licensing issues derived by dependencies between packages; they were able to find some licensing issues confirmed by Fedora. Manabe *et al.* [26] analyzed the changes in licenses of FreeBSD, OpenBSD, Eclipse, and ArgoUML, finding that each project had different evolution patterns. German *et al.* [19] analyzed the presence of cloned code fragments between the Linux Kernel and two distributions of BSD, i.e., OpenBSD and FreeBSD. The aim was to verify whether the cloning was performed in accordance to the terms of the licenses. Results show that, in most cases, these code-migrations were admitted.

While we share similar goals with prior related work—understanding insights into license usage and migration— our analysis is done on a much larger scale, i.e., across 16K projects *vs.* less than 10 projects in prior work (although German *et al.* [18] considered a single version of Fedora, the work investigated 1,475 source and 2,399 binary packages, which corresponded to each other, for that system). In addition, we performed an in-depth analysis of the rationale behind license usages and migrations by systematically studying and categorizing a multitude of related software artifacts (i.e., source code files, commit notes, and issue tracker discussions).

III. Design of Empirical Study

The *goal* of our study is to investigate license adoption and evolution in FOSS projects, with the *purpose* of understanding the overall rationale behind picking a particular license or changing licenses and of determining the underlying license change patterns. The *context* consists of the change history of 16,221 Java open source projects mined from GitHub, as well as their issue tracker discussions.

A. Research Questions

We aim at answering the following research questions:

1) **RQ**$_1$ *What is the usage of different licenses by projects in GitHub?* This research question examines the proportions of different types of licenses that are introduced by FOSS projects hosted in GitHub. In doing this, we should consider that GitHub is a relatively young forge, which has seen exponential growth in the number of projects over the past few years (see Table II), and that most of the projects it hosts are young in terms of the first available commit or the date that the repository was created.
2) **RQ**$_2$ *What are the most common licensing change patterns?* Our second research question investigates the popular licensing change patterns in the GitHub Open Source community with the aim of driving out—from a qualitative point of view—the rationale behind such change patterns.
3) **RQ**$_3$ *To what extent are licensing changes documented in commit messages or issue tracker discussions?* This research question investigates on whether licensing changes in a system can be traced to commit messages or issues' discussions.
4) **RQ**$_4$ *What rationale do these sources contain for the licensing changes?* This research question investigates the rationale behind the particular change in license(s) from a developer's perspective.

We address our four research questions by looking at the licensing phenomenon from two different points of view, namely (i) a *quantitative* analysis of the licenses under which projects were released, their changes across their evolution history, and the ability to match these changes to either commit notes or issue tracker discussions; and (ii) a *qualitative* analysis of developers' licensing-related discussions made over the issue trackers and of the way in which developers documented licensing changes through commit notes. For the case of licensing changes, we are interested in analyzing license migration patterns that fall in the following three categories:

- *No license $\rightarrow$ some License(s) – $N2L$*. This reflects the case where developers realized the need for a license and added a licensing statement to files;
- *some License(s) $\rightarrow$ No license – $L2N$*. In this case, for various reasons, licensing statements have been removed from source code files; for example, because a developer accidentally added a wrong license/license version;
- *some License(s) $\rightarrow$ some other License(s) – $L2L$*. This is the most general case of a change in licensing between distinct licenses.

With the *quantitative* analysis, we looked for answers to **RQ**$_1$, **RQ**$_2$, and **RQ**$_3$; whereas, with the *qualitative* analysis, we aimed at answering **RQ**$_4$.

B. Quantitative Analysis

In order to generate the data set to be used in the study, we mined the commit history of 16,221 Java Application publicly available on GitHub. GitHub hosts over twelve million *Git* repositories covering many popular programming languages, and provides a public API [3] that can be used to query and mine project information. Also, the *Git* version control system allows for local cloning of the entire repository, which facilitates the comprehensive analysis of the project change-history and thus of the license changes happened in each commit.

To extract data for our quantitative analysis, first we mined a comprehensive list of projects hosted on GitHub by implementing a script exploiting GitHub's APIs. GitHub limits the number of requests (or queries) per hour by IP Address to 60 for non-authenticated requests and 5,000 for the authenticated ones. The computation of the comprehensive list resulted in over twelve million projects. Since the infrastructure we use for license extraction supports Java systems (as it will be explained later), we filtered out all systems that were not written in Java, obtaining a list of 381,161 Java projects hosted on GitHub. We cloned all 381,161 git repositories locally for a total of 6.3 Terabytes of storage space. In our analysis, we randomly sampled 16,221 projects due to the computation time of the aforementioned infrastructure.

Once the *Git* repositories had been cloned, we used a code analyzer developed in the context of the MARKOS European project [9] to extract license information at commit-level granularity. The MARKOS code analyzer uses the Ninka license classifier [21] to identify and classify licenses contained in all the files hosted under the versioning system of each project. For each of the 16,221 projects in our study, the MARKOS code analyzer mined the change log, producing the following information for each commit:

1) *Commit Id:* The identifier of the commit that is currently checked out from the Git repository and analyzed;
2) *Date:* The timestamp associated with the commit;
3) *Author:* The person responsible for the commit;
4) *Commit Message:* The message attached to the commit;
5) *File:* The path of the files committed;
6) *Change to File:* A field to indicate whether each file involved in the commit was Added, Deleted, or Modified;
7) *License Changed:* A Boolean indicating value whether the particular file has experienced a change in license in this commit with respect to its previous version;
8) *License:* The name and version (e.g., *GPL-2.0*) of each license applied to the file.

The computation of such information for all 16,221 projects took almost 40 days, and resulted in the analysis of a total of 1,731,828 developers' commits involving 4,665,611 files. Note that for the BSD and CMU licenses Ninka was not able to correctly identify its version (reporting it as *BSD var* and

CMU var). Additionally, the GPL and the LGPL may contain a "+" after the version number (e.g., 3.0+), which represents a clause in the license granting the ability to use future versions of the license (i.e., the *GPL-2.0+* would allow for utilization under the terms of the *GPL-3.0*). Also, we have values of "no license" and "unknown", which represents the case that no license was attached to the file or Ninka was unable the determine the license.

We quantitatively analyzed the collected data by presenting descriptive statistics about the license adoption and the most common *atomic license changes* in the analyzed systems. The latter are defined as the commits in which we detected a specific kind of license change within at least one source code or textual file. For example, given a commit with three files experiencing the licensing change *No license* $\rightarrow$ *Apache-2.0*, and 10 files with *GPL-2.0* $\rightarrow$ *GPL-3.0*, the atomic license changes from that commit are one *No License* $\rightarrow$ *Apache-2.0* change and one *GPL-2.0* $\rightarrow$ *GPL-3.0* change. We prefer not to count the number of changes at file level as it was done in previous work [14] to avoid inflating our analysis because of large commits and to make comparable commits performed on both small and large projects. At the end, we identified a total of 1,833 projects with *atomic license changes* out of our dataset of 16,221 projects. This subset of projects was used to investigate license change traceability. Intuitively, we require the presence of license changes in order to determine how well changes in licensing are documented in either the commit notes or issue tracker discussion. Therefore, we used a web crawler to identify, among these 1,833 projects, those using the GitHub issue tracker, finding a total of 1,586 projects having at least one issue on it. To link the licensing changes to commit notes/issue reports, we performed both string matching and date matching between either the commit notes or the issue tracker discussions and the extracted licensing information (e.g., license name or date that license was committed).

C. Qualitative Analysis

Our qualitative analysis is based on manual inspection and categorization of commit notes and issue tracker discussions. While the former can be queried from the repository using *Git* commands, the latter have been extracted by building a web crawler collecting the information present in all issue trackers on GitHub of the 16,221 projects described above. In particular, for each issue our crawler collected (i) its title and description, (ii) the text of each comment added to it, (iii) and the date the issue was opened and closed (when applicable). In other words, we collected the discussion present in each issue. Of the 16,221 considered projects, 16,096 used the issue tracker (By "used," we refer to the existence of the issue tracker url. However, a subset of project's had a bug where their issue tracker redirected to pull-requests). In order to find the relevant issues (i.e., those presenting discussions about software licenses), we used a keyword search mechanism exploiting specific licensing keywords (e.g., *copyright*) or license names (e.g., *GPL*). In some cases, our keyword-filters included bi-grams composed by the license type and version, since some licenses types considered alone (e.g., *apache*) produced a very large amount of false positive discussions (e.g., all those talking about Apache projects). At the end, we obtained 273 issue-tracker discussions to manually analyze.

Concerning the commit notes, we premise that finding commit notes explicitly reporting the rationale behind a license adoption/change is far from trivial. Indeed, we found developers are very reluctant to document license changes in commit notes. Although we primarily considered 16,221 projects in our analysis (quantitative and traceability), we extracted commit notes from the complete commit history of 381,161 Java projects, which amounted to 63,564,326 commits, in order to perform our qualitative analysis. We sought to improve the ability to generalize our taxonomy by sampling from this complete history of all Java projects; as previously noted, it was infeasible to use the entire dataset for our qualitative analysis due to computation time. Among these commit messages, our keyword-based filter identified 742,671 commit messages (1.17%) likely talking about licenses. It is worth noting that we adopted a rather strict keyword-based filter based on the critical words exploited by Ninka during licenses identification augmented with license names. From these commits, we randomly sampled 500 total commit notes to understand the level of detail that commits contain with respect to licensing. In addition, we also considered 224 randomly sampled commit notes from the commits of the 1,833 projects in which we identified (in our quantitative analysis) an instance of an *atomic license change*.

After collecting commit notes and issue discussions, we used Grounded Theory (GT)—following the principles formulated by Corbin and Strauss [11]—to group them into categories. The GT-based classification of commits and issue tracker discussions aimed at finding the rationale for licensing changes in the analyzed dataset; in particular we aimed at answering the following two sub-questions: *What are the reasons pushing developers to associate a particular license to their project?* and *What causes them to migrate licenses or release their project under a new license (i.e., co-licensing)?*

For the GT-based analysis, we distributed the commit notes and the issue tracker discussions among the authors such that two authors were randomly assigned to each message (a message can be a commit note or an entire issue tracker discussion). After each round of *open coding*, in which the authors independently created classifications for the messages, we met to discuss the coding identified by each of us, and refined them into categories. Note that during each round the categories defined in previous rounds were refined accordingly to the new knowledge created from the additional manual inspections and from the authors' discussions. Overall, the GT analysis concerned (i) 500 randomly selected licensing-related commit notes identified via the keywords-based mechanism; (ii) the 224 commit notes from the commits where a licensing change was observed in our quantitative analysis; and (iii) the 273 issue tracker discussions matching licensing-related keywords. The output of our GT-based analysis is a set of categories and subcategories explaining why licenses are

TABLE I
TOP LICENSES: OSI, SOURCEFORGE, AND OUR DATASET.

OSI Popular License (unordered)	SourceForge (Dec. 2009)	Our Github Data Set
Apache-2 Lic	GNU Public Lics	GNU Public Lics
BSD 2-Clause Lic	Lesser GNU Public Lics	Apache Lics
BSD 3-Clause Lic	BSD Lics	Lesser GNU Public Lics
GNU Public Lics	Apache Lics	MIT Lic
Lesser GNU Public Lics	Public Domain	Eclipse Public Lic
MIT Lic	MIT Lic	Comm. Dev. and Dist. Lic
Mozilla Public Lic 2	Academic Free Lic	Mozilla Public Lic
Comm. Dev. and Dist. Lic	Mozilla Public Lics	BSD Lics
Eclipse Public Lic		

TABLE II
PROJECTS IN OUR DATASET WITH AN INITIAL COMMIT FOR EACH YEAR.

Year	Projects	Year	Projects	Year	Projects	Year	Projects	Year	Projects
1992	1	2000	7	2004	22	2008	186	2012	14159
1996	1	2001	11	2005	36	2009	263	2013	60
1997	3	2002	10	2006	72	2010	440		
1999	6	2003	35	2007	91	2011	811		

adopted and changed. We qualitatively discuss the findings of our GT-based analysis in Section IV-D, presenting our categories classification and examples of commit notes and issue tracker discussions belonging to the various categories.

D. Dataset Analysis

To have an idea of the external validity of our dataset, we measured the diversity metric proposed by Nagappan *et al.* [28] for our dataset. We matched the list of our mined projects from GitHub to the list of available projects from Boa [16], and ended up with 1,556 projects that were matched by name. This subset was used in the computation of the diversity metric, obtaining a score of 0.35, indicating that around 10% of our dataset covers just over a third of the open source projects according to six dimensions: programming language, developers, project age, number of committers, number of revisions, and number of programming languages. The dimensional scores are 0.45, 0.99, 1.00, 0.99, 0.96, 0.99, respectively, suggesting that our subset covers the relevant dimensions for our analysis. However, the focus on Java projects limits the programming language score, affecting the overall score.

Another important aspect to analyze is the representativeness of the licenses present in our dataset with respect to those diffused in the OS community. The Open Source Initiative (OSI) specifies a list of approved 70 licenses, indicating the ones reported in the first column of Table I as the most commonly used in FOSS software (they do not specify any order). The second column of Table I reports the top licenses as extracted from the FLOSSmole's SourceForge snapshot of December 2009 [23], while the third column shows the top licenses as extracted from our sample of GitHub projects.

The license declared by OSI as the most commonly used the most commonly found also in our dataset (BSD 2 and 3 fall both in the BSD type). In the comparison between our dataset and SourceForge, while the order of diffusion for the different licenses is not exactly the same, six of the top eight licenses in SourceForge are also present in our dataset (all but Public Domain and Academic Free License). This analysis, together with the diversity metric, suggests that our dataset is representative of OS systems.

Table II reports the year of the first commit date for each of the 16,221 considered projects. This table clearly shows the exponential growth of GitHub until 2012, confirming what already was observed by people in the GitHub community [15]. While GitHub also experienced exponential growth in 2013 [7], our dataset does not mirror this fact. This is due to a design choice we made while randomly choosing the projects to clone. In particular, we cloned projects during January 2014, excluding projects with a commit history less than one year from the set of 381,161 Java projects (i.e., projects with the first commit performed no later than January 2013). This was needed since, in the context of $\mathbf{RQ}_2$, we are interested in observing migration patterns occurring over the projects' history. Thus, projects having a very short commit history were not relevant for the purpose of this study. Moreover, since in $\mathbf{RQ}_1$ we are interested in observing licenses' usage in the context of the GitHub's drastic expansion, we decided to exclude the 60 projects having the first commit in 2013 from our analysis due to the severe lack of representation in our sample despite the continued growth of GitHub.

E. Replication Package

The working data set of our study is available at: http://www.cs.wm.edu/semeru/data/ICPC15-licensing.

IV. EMPIRICAL RESULTS

This section discusses the achieved results answering the four research questions formulated in Section III-A.

A. RQ_1: What is the usage of different licenses in GitHub?

Fig. 1 depicts the percentage of licenses that were first introduced into a project in the given year, which we refer to as *relative license usage*. We only report the first occurrence of each license committed to any file of the project. For easier readability, the bars are grouped by permissive (dashed bars) or restrictive licenses (solid bars). Additionally, we omit data prior to 2002 due to the limited number of projects created during those years in our sampled dataset (see Table II).

In 2002, we observed that restrictive licenses and permissive licenses have been used approximately equally with a slight bias toward using restrictive licenses. Although the *LGPL-2.1* and *LGPL-2.1+* variant are restrictive licenses, they are less restrictive than their GPL counter-part. It specifically aimed at ameliorating licensing conflicts that arose when linking code to a non-(L)GPL system; whereas, the GPL licenses would require the system to change its license to the GPL or else the component would not legally be able to be added. Thus, it suggests a bias toward using less restrictive licenses even among the typical copy-left licenses. By the subsequent year (2003), a clear movement toward using less restrictive licenses can be seen with the wider adoption of the *MIT/X11* license as well as the *Apache-1.1* license. Additionally, we observe that the *LGPL* is still prominent, while the *CMU*, *CPL-1.0*, and *GPL-2.0+* licenses were declining. During the following five years (2004-2008), the *Apache-2.0*, *CDDL-1.0*, *EPL-1.0*, *GPL-3.0*, *LGPL-3.0*, and *DWTFYW-2* licenses were created. Also during this period, the work of Bavota *et al.* showed that the Apache ecosystem grew exponentially [8]. This observation explains the rapid diffusion of the *Apache-2.0* license among FOSS projects. We observed a growth that resulted in the *Apache-2.0* license accounting for approximately 41% of licensing in 2008. Conversely, we observed a decline in the relative usage of both GPL and LGPL licenses, excluding 2007. Combined, the two observations suggest a stronger movement toward permissive licenses since approximately 65% of licenses attributed were permissive for 2005, 2006, and 2008; while initially it was at approximately 63% in 2004, the percentage of permissive licenses only dropped below 60% during 2007 to approximately 55%.

Another interesting observation was that the newer version of the GPL (*GPL-3.0* or *GPL-3.0+*) had a lower relative usage compared to its earlier version until 2011. Additionally, the adoption rate was more gradual than for the *Apache-2.0* license that appears to supersede *Apache-1.1* license. However, the *LGPL-3.0* or *LGPL-3.0+* does not have more popularity than prior versions in terms of adoption, despite the relative decline of the *LGPL-2.1*'s usage starting in 2010. Our manual analysis of commits highlighted explicit reasons that pushed some developers to chose the LGPL license. For instance, a developer of the `hibernate-tools` project when committing the addition of the *LGPL-2.1+* license to her project wrote:

> *The LGPL guarantees that Hibernate and any modifications made to Hibernate will stay open source, protecting our and your work*

This commit note indicates that *LGPL-2.1+* was chosen as the best option to balance the freedom for reuse and guarantee that the software will remain free.

Conversely, we observed the abandonment of licenses as newer FOSS licenses are introduced. For example, *Apache-1.1* and *CPL-1.0* become increasingly less prevalent or no longer used among the projects. In both cases, a newer license appears to replace the former license. While the *Apache-2.0* offers increased protections (e.g., protections regarding patent litigation), the *EPL-1.0* primarily resembles a textual replacement of "Common" in the *CPL-1.0* to "Eclipse" in the EPL as well as altering the copyright by replacing "IBM" with "The Eclipse Foundation". Thus, the two licenses are intrinsically the same from a legal perspective, which explains why the EPL adoption grew as the CPL usage shrunk.

Finally, we observed fluctuations in the the adoption of the *MIT/X11* license. As the adoption of permissive licenses grew with the introduction of the *Apache-2.0* license, it first declined in adoption and was followed by growth to approximately its original adoption. Ultimately, we observed a stabilization of the *MIT/X11* usage at approximately 10% starting in 2007.

Summary for RQ$_1$. We observed a clear trend towards using permissive licenses like *Apache-2.0* and *MIT/X11*. Additionally, the permissiveness or restrictiveness of a license seems to impact the adoption of newer versions, where permissive licenses are more rapidly adopted. Conversely, restrictive licenses seem to maintain a greater ability to survive in usage as compared to the permissive licenses, which become superseded. Finally, we observed a stabilization in the license adoption proportions of particular licenses, despite the exponential growth of GitHub.

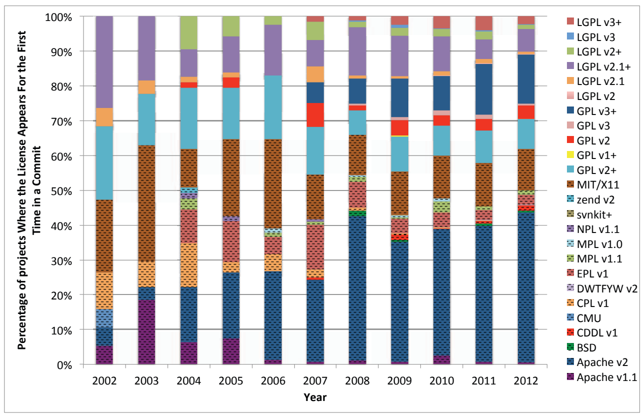

Fig. 1. Relative License Usage between 2002 and 2012 (dashed pattern representing permissive licenses).

B. RQ$_2$: What are the most common licensing change patterns?

We analyzed commits where a license change occurred, with a two-fold goal (i) analyze license change patterns to understand both the prevalence and types of changes affecting software systems, and (ii) understand the rationale behind these changes. Overall, we found 204 different *atomic license change* patterns. To analyze them, we considered their prevalence across the projects (i.e., global patterns) and within a project (i.e., local patterns). We sought to distinguish between dominant global patterns (Table III) and dominant local patterns (Table IV). The former was extracted by identifying and counting the presence of a pattern only once per project. The latter was extracted by first identifying and counting the patterns in a given project; then, those results were compared for each project to identify the patterns that were dominant in a local scope (i.e., within a given project).

The most dominant global patterns the projects were either a change from no license or an unknown license to particular license, or a change from a particular license to no license or an unknown license. By particular, we mean that we were able to extract the license. Table III shows the top 10 global patterns. We observe that the inclusion of *Apache-2.0* was the most common pattern for unlicensed or unknown code.

Table III also shows the most common global migrations when focusing the attention on migrations happened between different licenses. We observe that the migration toward the more permissive *Apache-2.0* was a dominant change among the top 10 *atomic license changes* for global license migrations. An interesting observation is the license upgrade and downgrade between *GPL-2.0+* and *GPL-3.0+*. *GPL-3.0* is considered by the Free Software Foundation as a compatible license with the *Apache-2.0* license. Due to the large usage of Apache code in Java, this pattern is quite expected. However, the migration *GPL-3.0+* $\rightarrow$ *GPL-2.0+* is interesting since still allows for the project to be redistributed as *GPL-3.0*, but allows for the usage as *GPL-2.0*, which is less restrictive, as well.

Table IV shows the most common local migrations. The migrations appear to be toward a less restrictive license or license version. The low frequency of the *atomic license change* local patterns indicates that migrating licenses is non-trivial. It can also introduce problems with respect to reuse. For example, we observed a single project where *GPL-1.0+* code was changed to *LGPL-2.0+* a total of 9 times. LGPL is less restrictive than GPL when the code is used as a library. Thus, if parts of the system are GPL, the developer must comply with the more restrictive and possibly incompatible constraints.

Until now, we considered *atomic license changes* among any file in the repository. This was needed since most of the analyzed projects lack of a specific file (e.g., license.txt) declaring the project license. To extract the declared project license, we considered a file in the top level directory named: *license, copying, copyright,* or *readme*. When just focusing on projects including such files, we extracted 24 different change patterns. Table V illustrates the top eight licensing changes between particular licenses (i.e., we excluded no license or unknown license from this table) for declared project licenses. We only considered the top eight, since there was tie between five other patterns or the next group of change patterns. We observe that the change from *Apache-2.0* → *MIT/X11* was the most prevalent license change pattern, and the co-license of *MIT/X11* with *Apache-2.0* is the second most prevalent one. Interestingly, this pattern was not dominant in our file-level analysis, although the GT analysis provided us support for this pattern. The *MIT/X11* license was used to allow commercial reuse, while still maintaining the project's Open Source nature.

Our third pattern of *GPL-2.0+* → *GPL-3.0+* in Table V was expected since it was tied for the most prevalent among global *atomic license changes*. Similarly, the patterns of *MIT/X* → *Apache-2.0*, *GPL-3.0+* → *Apache-2.0*, and *Apache-2.0* → *GPL-3.0* were also among the top eight global changes. Another notable observation is that license changes are frequently toward permissive licenses. Excluding the five changes from *Apache-2.0* → *GPL-3.0+*, the remaining changes for the top eight are either a licensing change from a restrictive (or copyleft) license to a permissive license or a licensing change between two different permissive licenses.

Summary for RQ$_2$. The key insight from the analysis of *atomic license change* patterns is that the licenses tend to migrate toward less restrictive licenses.

C. *RQ$_3$: To what extent are licensing changes documented in commit messages or issue tracker discussions?*

Table VI reports the results of the traceability linking between licensing changes and commit notes/issue tracker discussions. We found a clear lack of traceability between license changes in both the commit message history and the issue tracker. In both data sources, we first extracted the instances (i.e., commit messages and issue tracker discussions) where the keyword "license" appears **or** where a license name was mentioned (e.g., "Apache"). In the former case, we are identifying potential commits or issues that are related to licensing, while the latter attempts to capture those related to specific types of licenses.

TABLE III
TOP 10 GLOBAL ATOMIC LICENSE CHANGE PATTERNS.

Top Patterns (Overall)	Pattern Occurrences
no license or unknown → *Apache-2.0*	823
Apache-2.0 → no license or unknown	504
no license or unknown → *GPL-3.0+*	269
GPL-3.0+ → no license or unknown	181
no license or unknown → *MIT/X11*	163
no license or unknown → *GPL-2.0+*	113
GPL-2.0+ → no license or unknown	111
MIT/X11 → no license or unknown	98
no license or unknown → *EPL-1.0*	94
no license or unknown → *LGPL-2.1+*	91
Top Patterns Between Different Licenses	**Pattern Occurrences**
GPL-3.0+ → *Apache-2.0*	25
GPL-2.0+ → *GPL-3.0+*	25
Apache-2.0 → *GPL-3.0+*	24
GPL-2.0+ → *LGPL-2.1+*	22
GPL-3.0+ → *GPL-2.0+*	21
LGPL-2.1+ → *Apache-2.0*	16
GPL-2.0+ → *Apache-2.0*	15
Apache-2.0 → *GPL-2.0+*	13
MPL-1.1 → *MIT/X11*	11
MIT/X11 → *Apache-2.0*	11

TABLE IV
TOP 10 LOCAL ATOMIC LICENSE CHANGE PATTERNS BETWEEN DIFFERENT LICENSES.

Pattern	Pattern Occurrences
GPL-2.0+ → *GPL-3.0+*	36
GPL-2.0+ → *LGPL-3.0+*	15
LGPL-3.0+;*Apache-2.0* → *Apache-2.0*	12
GPL-3.0+;*Apache-2.0* → *Apache-2.0*	12
GPL-2.0+ → *LGPL-2.1+*	10
GPL-1.0+ → *LGPL-2.0+*	9
GPL-2.0+ → *GPL-3.0+*	9
GPL-3.0+ → *Apache-2.0*	8
GPL-3.0+ → *GPL-2.0+*	8
GPL-3.0+ → *LGPL-3.0+*	8

TABLE V
TOP 8 LICENSE CHANGE PATTERN IN A DECLARED LICENSE FILE OF A PROJECT (LICENSE,COPYING,COPYRIGHT, OR README FILE), EXCLUDING NO LICENSE OR UNKNOWN LICENSE.

Pattern	Pattern Occurrences
Apache-2.0 → *MIT/X11*	12
Apache-2.0 → *MIT/X11*;*Apache-2.0*	8
GPL-2.0+ → *GPL-3.0+*	7
MIT/X11 → *Apache-2.0*	6
GPL-3.0+ → *Apache-2.0*	6
MIT/X11;*Apache-2.0* → *Apache-2.0*	5
Apache-2.0 → *GPL-3.0+*	5
GPL-3.0+ → *MIT/X11*	3

TABLE VI
TRACEABILITY BETWEEN LICENSING CHANGES AND COMMIT MESSAGES OR ISSUE TRACKER DISCUSSION COMMENTS.

Data Source	Linking Query	Links
Commit Messages	Commits with the keyword "license"	70746
	Commits containing new license name	519
	Commits containing new license name and the keyword "license"	399
Issue Tracker Comment Matching	Comments from closed issues containing the keyword "license"	0
	Comments from closed issues containing the new license	0
	Comments from closed issues containing the new license and the keyword "license"	0
	Comments from open issues containing the keyword "license"	68
	Comments from open issues containing the new license	712
	Comments from open issues containing the new license and the keyword "license"	16
Issue Tracker Date-based Matching	Closed comments opened before license change and closed before or at license change	197
	Open comments open before the license change	2241
	Comments from closed issues open before the license change and closed before or at the license change with keyword "license"	0
	Comments from open issues open before the license change with keyword "license"	0
Issue and Commit Matching	Comments in closed issues containing the keyword "Fixed #[issue_num]"	66025
	Comments in open issues containing the keyword "Fixed #[issue_num]"	3407
	Comments in closed issues containing the commit hash where the license change occurs	0
	Comments in open issues containing the commit hash where the license change occurs	1

By using the first approach, we retrieved 70,746 commits and 68 issues, while looking for licenses' names we identified 519 commits and 712 issues. However, these numbers are inflated by false positives (e.g., "Apache" can relate to the license or it can relate to one of the Apache Foundation's libraries). For this reason, we then looked for commit messages and issue discussions containing both the word "license" as well as the name of a license. This resulted in a drop of the linked commit messages to 399 and in zero issue discussions. Such results highlight that license changes are rarely documented by developers in commit messages and issues.

We also investigated whether relevant commits and issues could be linked together. We linked commit messages to issues when the former explicitly mentions fixing a particular issue (e.g., "Fixed #7" would denote issue 7 was fixed). We observed that this technique resulted in a large number of pairs between issues and commits; thus, our observation of a lack of license traceability is not only an artifact of poor traceability for these projects. To further investigate the linking, we extracted the commit hashes where a license change occurred and attempted to find these hashes in the issue tracker's comments. Since the issue tracker comments contains the abbreviated hash, we truncated the hashes appropriately prior to linking. Our results indicated only one match for an open issue and zero matches for closed issues.

Finally, we attempted to link changes to issues by matching date ranges of the issues to the commit date of the license change. The issue had to be open prior to the change and if the issue had been closed the closing date must have been after the change. However, we did not find any matches with a date-based approach.

Summary for RQ$_3$. Both the issue tracker discussions and commit messages yielded very minimal traceability to license changes, suggesting that the analysis of licensing requires fine-grained approaches analyzing the source code.

D. RQ$_4$: What rationale do these sources contain for the licensing changes?

Given the limited traceability, we investigated both data sources to understand the quality of the rationale when licensing is mentioned in either the commit messages or the issue tracker discussions (as explained in our design). We used GT analysis to create a taxonomy for both data sources.

We initially sampled a filtered subset of 500 commit messages from the entire set of commit messages among all the Java projects and distributed it among the authors. We generated 19 categories from our open coding of the commit messages and the dominant ones were: *License Addition, Copyright Update, License Change,* and *False Positive*.

The *License Addition* category represented the commits that explained that a license, license file, or copyright information had been added to a project or file headers. This category has subcategories of *License Declaration/Specified* and *Generic Addition*. The latter represents a generic commit message of "Created LICENSE.md" that is automatically generated when a license is added to an existing GitHub project with GitHub's licensing feature. Similarly, this category also contains messages just stating "added license". The *License Declaration/Specified* sub-category includes commit messages reporting the exact license added (e.g., "Added GPL"). Such messages make it clear to any individual how the project or component is licensed, but they still do not explain the rationale behind the license choice.

Similarly, *License Change* has a subcategory of *License Declaration/Specified*, when the author specifies a new license. The commit message "Release to public domain" suggests that the project licensing has been changed to public domain. In our initial data for the commits, this category also could have been related to a *License Addition*. However, we also observed commits where a different license was chosen (e.g. "Switched to a BSD-style license") or the clauses of the license were modified (e.g. "The NetBSD Foundation has granted permission to remove clause 3 and 4 from their software").

Finally, we defined *False Positives* and *Unclear* to contain cases coded as unclear, unrelated, or non-informative. Unclear or not enough information signified messages that made it unable to determine the purpose of the commit from a licensing perspective. The message "But it still has it's license! So you can't use it exactly how you want!" indicates the presence of a license, but it does not indicate if it was pre-existing or just added. Additionally, it does not specify the license type and so for our purposes it does not offer any rationale.

For the subsequent round of analysis, we specifically targeted commit messages where a licensing change occurred so that we could understand the rationale behind the change. We did not apply a keyword for these messages since we knew they were commits related to changes in licensing. When reading these commits, we also included the *atomic license change* pattern that was observed at that particular commit to add context. We observed new support for the existing categories. We refer to new support as messages indicating new rationale for the existing categories. In addition to the new rationale, we also observed more support of our previous analysis, such as "Rewrite to get LGPL code." or "Changed license to Apache v2" for the category of *Licensing Change*.

In the case of *Licensing Removal*, we observed that licenses were removed due to code clean up, files deletion, and dependencies removal. For example, we observed the removal of the *GPL-2.0* license with the following commit message, "No more smoketestclientlib", indicating a removed, previously exploited, library. Additionally, licenses were removed as developers cleaned up the project (e.g. "More cleanup").

Fix Missing Licensing is related to a license addition, but it occurred when the author intended to license the file, but forgot in the initial commit or in the commit introducing the licensing. For example, one commit message noted, "Added missing Apache License header." This observation is important since it indicates that the available source code may inaccurately seem unlicensed.

An important observation from the second round of our analysis was the ambiguity of commit messages. For example, we observed a commit classified as *Copyright Update* stating, "Updated copyright info." However, this commit corresponded to a change in licensing from *GPL-2.0* to *LGPL-2.1+*. This case both illustrates the lack of detail offered by developers in commit notes, and it illustrates that an update can be more significant than adding a header or changing a copyright year.

Since our analysis of commits seemed to hit saturation, we finally investigated the issue tracker discussions for licensing related comments. The analysis of these discussions introduced six new categories: *License Clarification, Reuse, Choosing License, License Compatibility,* and *Contributor License Agreement*. *License Clarifications* represents the scenario where a non-contributor submits an issue to clarify project licensing or the implications of a license. This category demonstrates that licensing is not trivial when it comes to code reuse and developers are not always able to determine the license of a project. While it is related to reuse when the motivation for the question is to include the source code in another system, the category *Reuse* includes requests for a different license. For example, we found a developer comment on the issue thread stating, "I would love to use this library, but the lack of a license is prohibiting me from doing so." Similarly, developers wanting to include code for commercial use would request a re-license or dual-license of the MIT license.

One interesting observation is that developers also use the issue tracker to track the initial project licensing. To classify this scenario, we extracted the category *Choosing License*. For example, there was an issue titled "What license to use" that posed the question of "BSD, GNU GPL, APACHE?" This observation suggests that the issue tracker is also utilized as a discussion forum for a subset of projects.

Additionally, we identified the *License Compatibility* category from issues where a non-contributor identified an incompatibility in the licensing the project and the project's dependencies or where a non-contributor recommends a license-compatible library. The license incompatibility not only created a potential license violation for the project but also prevented the non-contributor from cataloging the system among projects hosted on F-Droid [2].

Finally, we identified a category of *Contributor License Agreement*. This scenario arises when a developer not initially on the project submits code to the project. In one case, a developer submitted a patch but it could not be merged into the system until that developer filled out the Contributor License Agreement (CLA). A CLA makes it explicit that the author of a contribution is granting the recipient project the right to reuse and further distribute such contribution [10]. Thus, it prevents the contributed code from being grounds for a lawsuit.

Summary for RQ$_4$. While our grounded theory analysis indicated some lack of documentation (e.g., prevalence of false positives) and poor quality in documentation with respect to licensing in both issue tracker discussion and commits messages, we formally categorized the available rationale. We also found that the rationale may be incomplete or ambiguously describe the underlying change (e.g., "Updated copyright info" representing a change between different licenses). Finally, we observed that issue trackers also served as conduits for project authors and external developers to discuss licensing.

V. Lessons and Implications

The analysis of the commit messages and *atomic license changes* highlighted a gap in the level of detail or information offered with respect to licensing. A developer interested in reusing code would be forced to check the source code of the component to understand the exact licensing or ask for clarification (using the issue tracker, for example). Additionally, the reason behind the change is not usually well documented. This detail is particularly important when a system uses external/third-party libraries since a license may change during the addition or removal of those libraries. An important observation from our GT analysis also stresses the need for better licensing traceability and aid in explaining the license grants/restrictions. We found several instances in which the issue tracker was used to ask for clarifications regarding licensing from external developers (i.e., not contributors) that sought to reuse the code; this seems to suggest that **code reuse is problematic for developers due to licensing. Therefore, our study demonstrates a need for clear and explicit licensing information for the projects hosted on a forge.**

The lack of traceability of licensing changes is also important for researchers investigating software licensing on GitHub. While we cannot generalize to other features, it does suggest that commit message analysis may be largely incomplete with respect to details of the changes made during that commit and ultimately source code analysis is necessary. One way to achieve this is developers can take advantage of summarization tools such as *ARENA* [27] and *ChangeScribe* [12], [24]. While *ARENA* analyzes and documents licensing changes at release level, *ChangeScribe* automatically generates commit messages; however, using *ChangeScribe* would require extending it to analyze licensing changes at commit level. Another option is that forges (and software tools in general) verify that every file contains a license and that every project properly documents its license (this feature could be optional). It would greatly improve traceability and assert a consistency among the repositories. In terms of licensing, it would be beneficial for developers using another project to be informed when a licensing change occurs. For example, a developer could mark specific projects as dependents and receive automated notifications when particular changes occur. This would be very beneficial with licensing since a change in the license of a dependency could result in license incompatibilities.

The GT analysis also suggests that commercial usage of code is a concern in the open source community. Currently, the MIT license seems to be the most prominent license for this purpose. Additionally, it is important to note that the license might be missing in a file shortly after it's initial commit. The lack of a license is an important consideration in open source development since it suggests that the code may in fact be closed source (or copyrighted by the original author). The aforementioned suggestion above would also serve to address this problem.

Finally, it seems that some projects may remain unlicensed until release. This observation suggests that licensing is important to developers, but it may not be considered relevant until the project is intended to be used by others. Thus, support for licensing existing projects would be beneficial in automating the declared license and licensing of the source code to prevent parts of a system from being unlicensed. This observation might also suggest that some projects in GitHub,are not intended to be open source.

VI. Threats to Validity

Threats to *construct validity* concern the relationship between theory and observation, and relate to possible measurement imprecision when extracting data used in this study. In mining the git repositories, we relied on both the GitHub API and the *git* command line utility. These are both tools under active development and have a community supporting them. Additionally, the GitHub API is the primary interface to extract project information. We cannot exclude imprecision due to the implementation of such API. In terms of license classification, we rely on *Ninka*, a state-of-the-art approach that has been shown to have 95% precision [21]; however, it is not always capable of identifying the license (15% of the time in that study). With respect to our developer rationale, we conducted a formal study using Grounded Theory. We distributed all of the data among two authors at each stage to ensure consistence and agreement of the classifications.

Threats to *internal validity* can be related to confounding factors, internal to our study, that could have affected the results. For the *atomic licensing changes*, we reduced the threat of having the project size as a confounding factor by representing the presences of a particular change at each commit. A license change typically is handled at a given instance and not frequently. By using commit-level analysis, we prevent the number of files from inflating the results so that they do not inappropriately suggest large numbers of changes occurred in a project. To analyze the changes across projects, we took a binary approach of analyzing the presence of a pattern. Therefore, a particular project would not dominate our results due to size.

Threats to *external validity* represent the ability to generalize the observations in our study. We do not claim that the rationale and *atomic license change* patterns are complete or consistent across all of the systems, especially projects written in other programming languages. Additionally, our data set is representative of only projects hosted on GitHub and written in Java so we do not claim that the results generalize to any Java project. GitHub's exponential growth and popularity as a public forge indicates that it represents a large portion of the open source community. While the exponential growth or relative youth of projects can be seen as impacting the data, these two characteristics represent the growth of open source development and should not be discounted. Additionally, GitHub contains a large number of repositories, but it may not necessarily be a comprehensive set of all open source projects or even all Java projects. However, the large number of projects in our dataset (and relatively high diversity metrics values as shown in Section III-D) gives us enough confidence about the obtained findings. Further evaluation of projects across other open source repositories and other programming languages would be necessary to validate our observations in a more general context. It is also important to note that our observations only consider open source projects. Since we need to extract licenses from source code, we did not consider any closed source projects and we cannot assert that any of our results would be representative in closed source projects.

VII. Conclusions

We empirically studied phenomena related to license usage and licensing changes in a set of 16,221 Java projects hosted on GitHub. Quantitative data automatically mined have been complemented with qualitative analysis manually performed on commit messages and issue tracker discussions to provide meaningful explanations to our findings, that are summarized as following:

- New license versions were quickly adopted by developers. Additionally, new license versions of restrictive licenses (e.g., *GPL-3.0* vs *GPL-2.0*) favored longer survival of earlier versions, unlike the earlier version of permissive licenses that seem to disappear;
- Licensing changes are predominantly toward or between permissive licenses, which ease some kind of derivative work and redistribution, e.g. within commercial products;
- On the one hand, developers post questions to the issue tracker to ascertain the project's license and/or the implications of the license suggesting that licensing is difficult;
- On the other hand, there is a clear, lack of traceability between discussions and related license changes.

This work is mainly exploratory in nature as it is aimed at empirically investigating license usage and licensing changes from both quantitative and qualitative points of view. Nevertheless, there are different possible uses one can make of the results of this paper. Our results indicate that developers frequently deal with licensing-related issues, highlighting the need for development in (semi)automatic recommendation systems supporting license compliance verification and management. Additionally, tools compatible or integrated within the forge to support licensing documentation, change notification, education (i.e., picking the appropriate license), and compatibility would benefit developers attempting to reuse code. While working in this direction, one should be aware of possible factors that could influence the usage of specific licenses and the factors motivating licensing changes. This paper provides solid empirical results and analysis of such factors from real developers.

As part of our future agenda, we are planning on extend our study to other forges and languages in order to corroborate our results. Also, we are planning on further investigating licensing issues across software dependencies.

Acknowledgements

This work is supported in part by the NSF CCF-1218129 and NSF CAREER-1253837 grants.

References

[1] The BSD 2-Clause License. http://opensource.org/licenses/BSD-2-Clause. Last accessed: 2015/01/15.

[2] F-Droid. https://f-droid.org/. Last accessed: 2015/01/15.

[3] GitHub API. https://developer.github.com/v3/. Last accessed: 2015/01/15.

[4] GNU General Public License. http://www.gnu.org/licenses/gpl.html. Last accessed: 2015/01/15.

[5] PF: The OpenBSD Packet Filter. http://www.openbsd.org/faq/pf. Last accessed: 2015/01/15.

[6] Software Package Data Exchange (SPDX). http://spdx.org. Llast accessed: 2015/01/15.

[7] State of the Octoverse in 2012 https://octoverse.github.com/. Last accessed: 2015/01/15.

[8] 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.

[9] 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.

[10] A. Brock. Project Harmony: Inbound transfer of rights in FOSS Projects. *Intl. Free and Open Source Software Law Review*, 2(2):139–150, 2010.

[11] J. Corbin and A. Strauss. Grounded theory research: Procedures, canons, and evaluative criteria. *Qualitative Sociology*, 13(1):3–21, 1990.

[12] L. F. Cortés-Coy, M. Linares-Vásquez, J. Aponte, and D. Poshyvanyk. On automatically generating commit messages via summarization of source code changes. In *Source Code Analysis and Manipulation (SCAM), 2014 IEEE 14th International Working Conference on*, pages 275–284. IEEE, 2014.

[13] 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.

[14] 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.

[15] B. Doll. The octoverse in 2012 http://tinyurl.com/muyxkru. Last accessed: 2015/01/15.

[16] R. Dyer, H. A. Nguyen, H. Rajan, and T. N. Nguyen. Boa: a language and infrastructure for analyzing ultra-large-scale software repositories. In *35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013*, pages 422–431, 2013.

[17] Free Software Foundation. Categories of free and nonfree software. https://www.gnu.org/philosophy/categories.html. Last accessed: 2015/01/15.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] J. Howison, M. Conklin, and K. Crowston. FLOSSmole: a collaborative repository for FLOSS research data and analyses. *IJITWE'06*, 1:17–26.

[24] M. Linares-Vásquez, L. F. Cortés-Coy, J. Aponte, and D. Poshyvanyk. ChangeScribe: A tool for automatically generating commit messages. In *37th IEEE/ACM International Conference on Software Engineering (ICSE'15), Formal Research Tool Demonstration*, page to appear, 2015.

[25] 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. ACM, 2010.

[26] 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.

[27] L. Moreno, G. Bavota, M. Di Penta, R. Oliveto, A. Marcus, and G. Canfora. Automatic generation of release notes. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, November 16 - 22, 2014*, pages 484–495, 2014.

[28] M. Nagappan, T. Zimmermann, and C. Bird. Diversity in software engineering research. In *Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE'13, Saint Petersburg, Russian Federation, August 18-26, 2013*, pages 466–476, 2013.

[29] Oracle. MySQL - FOSS License Exception. http://www.mysql.com/about/legal/licensing/foss-exception/. Last accessed: 2015/01/15.

[30] P. Sing and C. Phelps. Networks, social influence, and the choice among competing innovations: Insights from open source software licenses. *Information Systems Research*, 24(3):539–560, 2009.

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