**Noname manuscript No.**
(will be inserted by the editor)

# License Usage and Changes: A Large-Scale Study on GitHub

**Christopher Vendome,**
**Mario Linares-Vásquez,**
**Gabriele Bavota,**
**Massimiliano Di Penta,**
**Daniel German,**
**Denys Poshyvanyk**

Received: date / Accepted: date

**Abstract** Open source software licenses determine, from a legal point of view, under which conditions software can be integrated and redistributed. The reason why developers of a project adopt (or change) a license may depend on various factors, *e.g.,* the need for ensuring compatibility with certain third-party components, the perspective redistribution or commercialization of the software, or the need for protecting against somebody else's commercial usage of software. This paper reports a large empirical study aimed at *quantitatively* and *qualitatively* investigating when and why developers adopt or change software licenses. Specifically, we first identify licenses' changes in 39,563,885 commits, representing the entire history of 51,757 projects hosted on GitHub written in C, C++, C#, JavaScript, Python, and Ruby. Then, to understand the rationale of license changes, we perform a qualitative analysis—following an open coding approach inspired by grounded theory—on commit notes and issue tracker discussions concerning licensing topics and, whenever possible, try to build trace-

C. Vendome
The College of William and Mary
E-mail: cvendome@cs.wm.edu

M. Linares-Vásquez
The College of William and Mary
E-mail: mlinarev@cs.wm.edu

G. Bavota
Free University of Bolzano
E-mail: gabriele.bavota@unibz.it

M. Di Penta
University of Sannio
E-mail: dipenta@unisannio.it

D. M. German
University of Victoria
E-mail: dmg@cs.uvic.ca

D. Poshyvanyk
The College of William and Mary
E-mail: denys@cs.wm.edu

ability links between discussions and changes. On one hand, our results highlight how, in different contexts, license adoption or changes can be triggered by various reasons. On the other hand, the results also highlight 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. In conclusion, results of the study trigger the need for a better tool support for guiding developers in choosing/changing licenses, and in keeping track of the rationale behind license changes.

**Keywords** Software Licenses · Mining Software Repositories · Empirical Studies

## 1 Introduction

In recent and past years, the availability of Free and Open Source Software (FOSS) projects is significantly increasing, along with the availability of forges hosting such projects (*e.g.,* SourceForge[1] or GitHub[2]) and foundations supporting and promoting the development and diffusion of FOSS (examples are the Apache Software Foundation[3], the GNU Software Foundation[4], or the Eclipse Software Foundation[5]). The availability of open source software 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. Noteworthy, this can happen not only in the context of open source projects; such practices are becoming more frequent even in commercial projects.

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. In order to license a software project, developers either add a *licensing statement* as a comment on top of source code files, or else include a textual file containing the license statement in the project source code root directory or in its sub-directories.

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 [29, 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

[1] `http://sourceforge.net`
[2] `https://github.com`
[3] `https://www.apache.org`
[4] `http://www.gnu.org`
[5] `http://www.eclipse.org/`

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.,* towards 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 [28] to allow the driver's inclusion in other software released under some open source licenses in principle incompatible with the GPL (*e.g.,* the Apache license). 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. Also, there are some specific Web applications (*e.g.,* `http://choosealicense.com/`) helping developers to choose a license. Furthermore, 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 [32].

**Paper contributions.** This paper reports the results of a large empirical study aimed at quantitatively and qualitatively investigating when and why licenses change in projects, which were written in C, C++, C#, JavaScript, Python, and Ruby, hosted on GitHub. We also compare the findings to our earlier work on 16,221 Java projects [33]. To conduct this study, we first mined the entire change history of 4,671 C, 1,902 C#, 4,209 C++, 14,161 JavaScript, 9,349 Python, and 17,984 Ruby projects extracting the license type (*e.g.,GPL* or *Apache*) and version (*e.g.,v1*, *v2*) from each of the 32,223,454 files involved in a total of 39,563,885 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. After that, following an open coding approach inspired by grounded theory [11], we qual-

itatively analyze a sample of commit messages and issue tracker discussions likely related to license changes. Such qualitative analysis allowed us to provide a rationale on the reasons why developers adopt specific license(s), both for initial licensing and for licensing changes. We discuss such results along with those previously obtained for Java projects [33].

The study reported in this paper poses its basis on previous work aimed at exploring license incompatibilities [18], license changes [14], license evolution[25] and integration patterns [20]. Building upon previous work on licensing analysis, this paper make the following specific contributions:

1. To the best of the authors' knowledge, this paper represents 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] as well as 16,221 Java projects in our earlier work [33], whereas this work analyzes 51,757 projects) and considers multiple popular programming languages;
2. To the best of our knowledge, this 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 achieved 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. We also observe that licensing expectations may be different based on the programming language. Although choosing a license is considered important for developers, even since early releases of their projects, forges and third party-tools provide little or no support to developers when performing licensing-related tasks, *e.g.,* picking a license, declaring the license of a project, changing license from a restrictive one towards a more permissive one (or *vice versa*) and, importantly, keeping track of the rationale for license changes. Also, there is a lack of consistency 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.*). 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. Last, but not least, the currently available Software Configuration Management (SCM) technology provide a very limited support to trace licensing-related discussions and decisions onto actual changes, whereas such traceability links can be useful to understand the impact of such decisions.

**Paper structure.** The paper is organized as follows. Section 2 overviews existing work on licensing analysis. Section 3 describes the study design and the details behind the data analysis procedure. The results are reported and discussed in Section 4. Lessons learned from the study results are summarized in Section 5, while Section 6 discusses the threats to the study's validity. Finally, Section 7 concludes the paper and outlines directions for future work.

## 2 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.

### 2.1 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.* [30] proposed 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 and version. In the evaluation reported by the authors, Ninka achieved a precision $\sim$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 may 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. Note that, differently from the previous techniques, the approach in [13] is only able to identify the license family (*e.g.,* GPL) without specifying the version (*e.g.,* 2.0).

### 2.2 Empirical Studies on License Adoption and Evolution

Di Penta *et al.* [14] investigated—on six open source projects written in Java, C, and C++ programming languages—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.

Manabe *et al.* [25] analyzed the changes in licenses of FreeBSD, OpenBSD, Eclipse, and ArgoUML, finding that each project had different evolution patterns.

With respect to the above works, we (i) perform a larger study on projects written in five different programming languages (six considering our previous results for Java [33]), and (ii) complement the quantitative analysis with open coding of licensing-related commit messages and issue tracker discussions.

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.

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 since they went from less restrictive licenses towards more restrictive ones.

Wu *et al.* [34] investigated license inconsistencies between cloned files. They performed an empirical study on Debian 7.5 to demonstrate the ways in which licensing can become inconsistent between the file clones (*e.g.,* the removal of a license in one of the clone pairs).

Vendome *et al.* conducted a survey with developers that contributed to projects that had experienced changes in licensing to understand the rationale for adopting and changing licensing [32]. The survey results indicated that facilitating commercial reuse is a common reason for license changes. Also the survey highlighted that, in general, developers lack understanding of the legal implications of open source licenses, which highlights the need for recommenders aimed at supporting them in choosing and changing the licenses.

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 51K+ projects (besides the 16K from the original study to which we compare the new findings [33]) *vs.* less than ten 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 for that system). This work also investigates multiple programming languages to understand the influence that a programming language may have with respect to licensing. 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).

## 3 Design of the Empirical Study

The *goal* of our study is to investigate license adoption and evolution in FOSS projects hosted on GitHub, with the *purpose* of understanding the rationale behind adding licensing, picking a particular license, or changing licenses. The *perspective* is of researchers interested in understanding what are the main factors leading towards specific license adoption and change.The *context* consists of the change history of 17,984 Ruby, 14,161 JavaScript, 9,349 Python, 4,671 C, 3,690 C++, and 1,902 C# open source projects mined from GitHub, as well as their issue tracker discussions.

This study is a replication of our previous work, where we limited our attention to the analysis of 16,221 Java projects [31].

### 3.1 Research Questions

Our study aims at answering the following four 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, and that most of the projects it hosts are young in terms of the first available commit or the date when 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 (*e.g.,* satisfying dependency constraints).
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 issue 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 the 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.

While **RQ**$_1$ and **RQ**$_2$ have been addressed by performing a quantitative analysis of licensing contained in the source code files and their changes, **RQ**$_3$ and **RQ**$_4$ have been addressed through a *qualitative* analysis of commit messages and issues posted on the projects' issue trackers. Since the qualitative analysis requires multiple rounds

of manual classification of artifacts, as it will be detailed below, it is limited to a sample of 1,133 commits and 213 issue discussions from 857 projects (189 projects contributing with issue discussions, 685 projects with commits, and 17 projects with both).

### 3.2 Context selection

In order to extract the data set to be used in the study, we mined the commit history of a total of 51,757 projects, including 17,984 Ruby projects, 14,161 JavaScript projects, 9,349 Python projects, 4,671 C projects, 3,690 C++ projects, and 1,902 C# projects 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 detected 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. Our script exploited authenticated tokens from personal accounts to maximize our request limit and utilized timeouts to ensure compliance with GitHub's regulations.

The computation of the comprehensive list resulted in over twelve million projects. We excluded the Java projects, since we considered a significant number of projects in this programming language in our previous study [31] that this study replicates. We focused on systems written in six programming languages: C, C++, C#, Python, JavaScript, and Ruby by selecting only those satisfying the following two criteria: (i) they were not forks of the main repository, and (ii) they had at least one star (*i.e.,* at least one user expressed appreciation for the repository) or watcher (*i.e.,* at least one user asked to receive notification about changes made in the repository). These selection criteria were used to exclude from our analysis personal repositories (*e.g.,* the website of a person) that might have biased our results. All 51,757 projects satisfying the selection criteria were cloned in a workstation.

### 3.3 Quantitative analysis of licensing and their changes

Once the *Git* repositories had been cloned, we used a code analyzer developed in the context of the MARKOS European project [9] to extract licensing 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 51,757 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 value indicating whether the file has experienced a change in a license in this commit with respect to its previous version. This feature applies to modified files only. In the case of an addition or deletion of a file, this field is set to *false*;
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 51,757 projects took almost 60 days, and resulted in the analysis of a total of 39,563,885 developers' commits involving 32,223,454 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 analyze the collected data by presenting descriptive statistics about the license adoption and the most common *atomic license changes* in the analyzed projects. 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 ten 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 commits performed on both small and large projects comparable.

At the end, we did not identify any license changes among the six languages. We investigated the repositories as well as the number of commits to better understand our dataset. We observed that the vast majority of projects were smaller in terms of the number of commits. Additionally, we observed that certain projects appeared to be mirrored on GitHub, indicating that they were mature but their development took place elsewhere. We also observed that certain projects had commits prior to the starting commit of the repository so it is possible that segments of the history were inaccessible to our analysis. Thus, we observe that many projects are either immature and so they do not have enough change history, where licensing changes are likely to occur, and it would be supported by the findings in [32]. Alternatively, the projects may be mature, but the change histories are not complete. However, we do present the findings from the Java study to which we compare our work. For Java, we

identified *atomic license changes* for 1,833 out of 16,221 Java 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., a license name or a date when a license was committed).

### 3.4 Qualitative Analysis

Our qualitative analysis is based on manual inspection and categorization of a sample of issue tracker discussions and commits related to licensing changes. While commits can be queried from the repository using the *Git log* command, the former have been extracted by building a Web crawler collecting the information present in all issue trackers of the studied projects. 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 order to find the relevant issues (*i.e.,* those presenting discussions about software licenses), we used a keyword search mechanism aimed at matching, in the issue title, specific licensing keywords reduced from Ninka's list (*e.g., copyright*) or license names (e.g., *GPL*). Specifically, we considered the following case-insensitive keywords:

> *copyright, compliance, gpl, gpl-2, gpl-3, gplv2, gplv3, gplv2+, gplv3+, lgpl, lgpl-2, lgpl-2.1, lgpl-3, lgplv2, lgplv2.1, lgplv3, lgplv2+, lgplv2.1+, lgplv3+, licenses, license, licensed, licensee, lgpl, merchantability, mit/x-derivative, mpl, written permission, prior permission, see the copyright.txt, licensing, licencing, liability, legal, public domain, special exception, copyright holders, to permit this exception, disclaims copyright, gpl, apache-2, apache-2.0, apache 2, apache 2.0, apache v2, apache v2.0, apache-1.1, apache 1.1, apache v1.1, apl-1.1, apl-1.1, apl 1.1, apl v1.1, gpl 3, gpl 3+, gpl 2, gpl 2+, lgpl 2, lgpl 2+, lgpl 2.1, lgpl 2.1+, lgpl 3, lgpl 3+, gpl v3, gpl v3+, gpl v2, gpl v2+, lgpl v2, lgpl v2+, lgpl v2.1, lgpl v2.1+, lgpl v3, lgpl v3+, mit/x, mit/x11, mit x11, mit expat, cpl-1.0, cpl-1, epl-1.0, epl-1, cpl 1.0, cpl 1, epl 1.0, epl 1, cddl-1.0, cddl-1, cddl 1.0, cddl 1, cpl v1.0, cpl v1, epl v1.0, epl v1, cddl v1.0, cddl v1, mpl-1.0, mpl-2.0, mpl-1, mpl-2, mpl 1.0, mpl 2.0, mpl 1, mpl 2, mpl v1.0, mpl v2.0, mpl v1, mpl v2, bsd-3, bsd-2, bsd-4, bsd 3-clause, bsd 2-clause, bsd 4-clause*

In some cases, our keyword-filters included bi-grams composed of 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). In the end, we identified a total of 213 issue tracking discussions potentially

related to licensing, including 79 from JavaScript projects, 45 from Ruby projects, 41 from Python projects, 30 from C projects, 12 from C++ projects, and 6 from C# projects. We compare these findings to our previous work [31], where we analyzed 273 issues from Java projects.

To identify commit notes likely related to license changes, we adopted a keyword-based filtering based on the critical words exploited by Ninka during licenses identification augmented with license names. Specifically, we used the following case-insensitive keywords:

> *as is, wrote this file, acknowledgement, advertising, agreement, attribution, authorization, compliance, conditions, copies, being used are not cryptographic, damages, derivative, disclaimed, disclaimer, distribute, distributed, distributing, free distribution, embargoed, executable file, fee, fees, redistributions in any form, redistributions of any form, this can be in the form of a textual message, free software, furnished, gpl, gpl-2, gpl-3, gplv2, gplv3, gplv2+, gplv3+, lgpl, lgpl-2, lgpl-2.1, lgpl-3, lgplv2, lgplv2.1, lgplv3, lgplv2+, lgplv2.1+, lgplv3+, grant, granted, http://www.gnu.org/licenses/, for more details, juridiction, jurisdiction, law, legend, licenses, license, licensed, licensee, lgpl, merchantability, mit/x-derivative, misrepresented, mpl, notice, obligation, written permission, prior permission, product includes, particular purpose, redistribute, redistribution, reexported, reproduce, use this software, restriction, all rights, royalty, see the revision, see the included, see the copyright.txt, for details, all intellectual property rights, sale, sell, subject to, terms, warranties, warranty, licensing, licencing, liability, meet some day, notices, legal, accompanying, included with this distribution for more information, See the file, public domain, special exception, notwithstanding, copyright holders, to permit this exception, suitability, computer program whose purpose, disclaims copyright, software is covered, Copyright*

In the end, the keyword-based filtering allowed us to identify a total of 4,203 commits, including 1,133 from C, 148 from C#, 694 from C++, 650 from Python, 607 from JavaScript, and 971 from Ruby projects. Given the high number of relevant commits, we sampled 20% of the commits found for each language as object of our manual inspection. However, we set a minimum threshold of 100 commits per language (*e.g.,* in case of C projects, despite the 20% of 148 being 30 commits, still we analyzed 100 randomly selected commits). This minimum threshold was adopted to ensure representativeness for each of the studied languages. Note that our sampling is statistically significant with a 95% confidence interval $\pm$ 10% or better. The number of sampled commits by language is as the following:

- C: 227 commits out of 1,133;
- C#: 100 commits out of 148;
- C++: 139 commits out of 694;
- Python: 130 commits out of 650;
- JavaScript: 122 commits out of 607;
- Ruby: 195 commits out of 971.

After collecting commit notes and issue discussions, we used an open coding procedure inspired by the Grounded Theory (GT) [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, the authors met to discuss the coding identified by each of the authors, and refined the messages into the 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 discussions among the authors.

Overall, the GT-based analysis concerned (i) 1,133 randomly selected licensing-related commit notes identified via the keyword-based mechanism; and (ii) the 213 issue tracker discussions where the title matched licensing-related keywords. The output of our GT-based analysis is a set of categories explaining why licenses are adopted and changed. We qualitatively discuss the findings of our GT-based analysis in Section 4.4, presenting our categories classification and examples of commit notes and issue tracker discussions belonging to the various categories. We compared the classification to our prior taxonomy from the study of Java projects.

### 3.5 Dataset Analysis

To assess external validity of our dataset, we measured the diversity metrics proposed by Nagappan *et al.* [27] 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 4,413 projects that were matched by name. This subset was used in the computation of the diversity metric, obtaining a score of 0.39, indicating that our dataset covers two fifths 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 0.47, 0.99, 1.00, 0.99, 0.98, 0.99, respectively, suggesting that our subset covers the relevant dimensions for our analysis. However, the focus on six programming languages inherently limits the programming language score (as these are a subset), which affects 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 1 as the most commonly used in FOSS software (they do not specify any order). The second column of Table 1 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.

**Table 1** Top licenses: OSI, SourceForge, and our dataset.

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

**Table 2** Projects in our dataset with an initial commit for each year.

| Year | Projects | Year | Projects | Year | Projects | Year | Projects | Year | Projects |
|---|---|---|---|---|---|---|---|---|---|
| 1980 | 1 | 1995 | 5 | 2001 | 77 | 2007 | 1,187 | 2013 | 4,565 |
| 1988 | 2 | 1996 | 4 | 2002 | 78 | 2008 | 10,119 | 2014 | 68 |
| 1990 | 3 | 1997 | 17 | 2003 | 99 | 2009 | 11,192 | 2015 | 15 |
| 1991 | 1 | 1998 | 22 | 2004 | 112 | 2010 | 554 | | |
| 1993 | 5 | 1999 | 37 | 2005 | 247 | 2011 | 9,168 | | |
| 1994 | 1 | 2000 | 49 | 2006 | 572 | 2012 | 11,616 | | |

The license declared by OSI as the most commonly used was also the most commonly found in our dataset (BSD 2 and 3 fall both in the BSD type). In 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 metrics, suggest that our dataset is representative of open source systems.

Table 2 reports the year of the first commit date for each of the 16,221 considered projects. This table clearly shows the large growth of GitHub around 2008, 2009, 2011, 2012, and 2013, which demonstrates that the projects conform to the rapidly growing forge. However, we observe that the projects are older confirming what already was observed by people in the GitHub community [15]. GitHub also experienced exponential growth in 2013 [7], however, our dataset does not mirror this fact for any year after 2012. In particular, we cloned the projects during July 2015. This was needed since, in the context of $\mathbf{RQ}_2$, we are interested in observing migration patterns occurring over the projects' change history. Thus, projects having a very short change history were not relevant for the purpose of this study. Moreover, since in $\mathbf{RQ}_1$ we are interested in observing license usages in the context of the GitHub's drastic expansion, we decided to exclude the 4,608 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.

### 3.6 Replication Package

The working data set of our study is available at: `http://www.cs.wm.edu/semeru/data/IEMSE15-licensing`. It includes the lists of projects and their urls, the issues tracker and commit data, analysis scripts, and results.

## 4 Study Results

This section discusses the achieved results answering the four research questions formulated in Section 3.1.

### 4.1 $RQ_1$: What is the usage of different licenses in GitHub?

Figures 1, 2, 3,4, 5, 6, and 7 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 2).

Figure 2 shows the license usage results for C projects. Overall, we see a predominant usage of restrictive licensing. In 2002, we observed that restrictive licenses accounted for approximately 87% of the licenses used in C projects. *GPL-2.0+* and *LGPL-2.0+* were the most prevalent licenses utilized, but *LGPL-2.1+* starts to become more prevalent in 2004 and stays close in size to *LGPL-2.0+*. Overall, *GPL-2.0+* remains the most prevalent license through the period of time between 2002 and 2012. In 2007, *GPL-3.0+* starts to be utilized in the projects and diminishes the share of *GPL-2.0+* usage. Interestingly, we do not observe a clear trend toward utilizing less restrictive copy-left license (i.e., the *LGPL* family of 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. This behavior opposes our findings in Java projects as seen in Figure 1, which suggested a bias toward using less restrictive licenses even among the typical copy-left licenses. While restrictive licenses remain the most commonly used in C projects, we do observe a a wider adoption of the *MIT/X* license and to a much smaller extent the *Apache-2.0* license. One explanation for the adoption of the *Apache-2.0* is compatibility with the *GPL-3.0(+)* license. Until 2011, we observed that restrictive licenses represented approximately at least 70% or more of license usage (excluding 2008, which drops to approximately 67%). In 2012, it drops further to 60% usage of restrictive licensing.

Additionally, we observe a similar phenomena to the Java license usage in that restrictive licenses seem to survive the introduction of newer licenses. We observe a consistent presence of *GPL-2.0* and *GPL-2.0+* licenses. While the proportion diminishes to greater usage of *GPL-3.0+* and *MIT/X* licenses, *GPL-2.0+* in particular

remains the most prevalent license. Thus, we do not observe the same behavior that older versions of restrictive remain widely used despite newer versions.

Figure 3 shows the license usage for C++ projects and reflects a similar behavior as C projects. In 2002, all licenses were restrictive licenses. In 2003, *MIT/X* is introduced as the only permissive license, but it is not adopted the following year and so 2004 only depicts restrictive license adoption. Starting in 2005, we observe that C++ projects begin following a more similar behavior to C projects, where permissive licenses become more prevalent led by *MIT/X* and *Apache-2.0*. However, *Apache-2.0* reaches approximately the same usage proportion as *MIT/X*. As previously stated, the *Apache-2.0* usage may be related to compatibility to *GPL-3.0(+)*. Additionally, this seems more likely as the *Apache-2.0* usage coincided with less *GPL-2.0+* usage (an incompatible license). Until 2010, we observe that restrictive licenses account for at least about 75% of the licenses introduced each year. While we initially observe *GPL-2.0+* as the most commonly introduced license (similar to C), we observe more diffused license usage among the restrictive licenses. Similarly, we observe the same survivability of restrictive licenses.

In Figure 4, we observe that 2002 and 2003 did not have any licenses being introduced into C# projects. Interesting, we observe sufficiently more volatility in licensing. By that, we mean that the license usage drastically changes each year. This behavior is likely due to the small number of detected license adoptions in the period of time between 2002 and 2012. Initially, we observe *GPL-2.0* and *GPL-2.0+* equally sharing the prevalence for license usage. In 2005, *LGPL-2.1(+)* is most widely introduced into the C# projects. In 2007, permissive licenses with the *MIT/X* license accounts for 60% of the licenses that were introduced. Between 2008 and 2012, we observed fluctuations where permissive and restrictive licensing alternates as the most dominant type of license. In this time, we also observe a wider diversity of licenses being introduce as compared to the prior years. Additionally, the *MIT/X* license is the single most prevalent license starting in 2007. This observation is similar to the observations in JavaScript and Ruby described later in this section.

Similarly to the C#, Figure 5 shows the lack of license introduction in 2002 and 2004, and it shows the consistent movement from restrictive to predominantly (almost entirely) permissive licenses as well the dominance of the introduction of the *MIT/X* license. Similar to C#, JavaScript had a limited number of projects with licenses introduced; however, it is interesting that *MPL-1.1* had such a large prevalence (almost 30%) in 2004, since its relative usage is sufficiently less in other languages. More importantly, we observe a rapid movement towards permissive licenses. By 2008, over 70% of the licenses introduced were permissive licenses. It decreases in 2009 and 2010, where permissive licenses are less prevalent; however, the trend reversed by 2012 and we observed that permissive licenses were approximately 90% of the licenses that were being adopted by JavaScript projects. By 2012, *MIT/X* maintained a large prevalence, but *Apache-2.0* also accounts for a large proportion of licenses introduced at approximately 39%. Thus, we observed a much stronger movement towards permissive licensing than C# projects.

Interestingly, Figure 6 demonstrates that restrictive licenses are more prevalent than permissive licenses until 2011. In 2011 and 2012, we observe that permissive licenses are more prevalent predominantly due to *Apache-2.0* and *MIT/X* licenses.

We first observe an increase in restrictive license adoption as it reaches 100% in 2003. Subsequently, it declines in 2004 with the adoption of *MPL-1.1*, but reclaims dominance in 2005 as *MPL-1.1* is not introduced to any projects that year. In 2006, we observe the beginning of a decline for restrictive licensing. *MIT/X* and *Apache-2.0* start gaining prevalence among the introduced licensing, and permissive licenses adoption exceed the adoption of restrictive licenses in 2011. Additionally, we observe a similar phenomena as seen in C and C++ projects, where developers still adopt earlier versions of restrictive licenses (both the *GPL* and *LGPL* families of licenses). In fact, we observe that despite *GPL-3.0(+)* becoming more prevalent and *GPL-2.0(+)* declining, the two licenses are adopted with a similar (almost equal) frequency in 2011 and 2012.

In the case of Ruby projects, we observe that the *MIT/X* license is the most prevalent license during the entire period of time between 2002 and 2012, as seen in Figure 7. In 2002 and 2003, we only observe the *MIT/X* license, and *GPL-2.0+* is the only license introduced in 2004 and 2005. In 2006, restrictive licenses reach their greatest proportion of adoption at approximately 35% with the additional adoption of *GPL-2.0*. However, we observe permissive licenses becoming more prevalent starting in 2007 as *Apache-2.0* is introduced more to the projects in our dataset. In 2011 and 2012, permissive licenses account for over 90% with approximately 74% of the license adoptions being the *MIT/X* license and 12% being the *Apache-2.0* license with the remaining few percent distributed among *MPL-1.1*, *MPL-1.0*, *DWTFYW-2.0*, and *BSD-variant* licenses. It is important to not that unlike C# and JavaScript, Ruby had a sufficiently larger number of projects adopting licenses (e.g., the adoption in 2008 of the *MIT/X* license represents 1,608 projects adoption the *MIT/X* license).

Comparing the aforementioneds findings of the six languages of our study to Java projects, Figure 1 demonstrates that initial restrictive licenses were slightly more prevalent than permissive licenses across the Java projects. 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.

Fig. 1 shows the initial results for Java projects. 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.

Interestingly, we observed a much stronger prevalence of *MIT/X* licenses in our dataset as compared to Java, where *Apache-2.0* was more prevalent. It is likely that external factors like community influence, such as the Apache Foundation, contributes the observations and would corroborate the results from [32]. The findings also parallel the issue tracker discussions, where we observed that developers request the usage of permissive licenses, like the *MIT/X* license, to facilitate commercial reuse.

The results also suggest that the language may impact the license usage and trends. For example, Java leans towards the adoption permissive *Apache-2.0* licensing, while C projects typically adopted more restrictive *GPL* and *LGPL* families of licenses, and Ruby projects predominantly adopted the *MIT/X* license. Additionally, we observed less licensing diversity across the projects, excluding C projects. In fact, the license diffusion tended to be biased more towards a few licenses in our dataset as compared to Java, where we observed a large diversity. Since we observed more adoption of *MIT/X* until the introduction of *Apache-2.0*, we do not observe similar patterns of newer permissive licenses replacing their earlier versions in terms of adoption. However, the results of the other languages further demonstrate the survival of

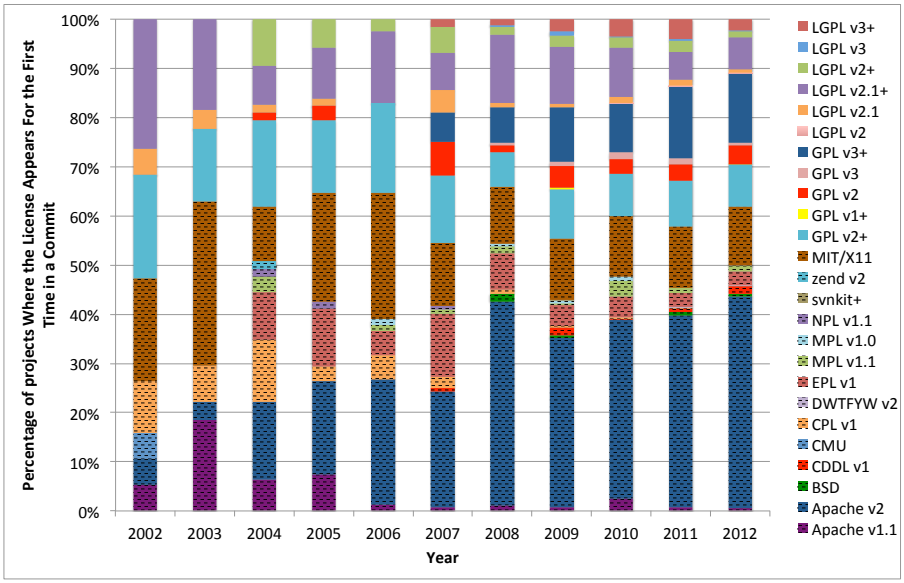


**Fig. 1** Relative License Usage between 2002 and 2012 for Java Projects (dashed pattern representing permissive licenses).

restrictive licenses. In particular, we observe that *GPL-2.0* remains widely adopted despite the release of the *GPL-3.0* license.

**Summary for RQ$_1$.** We observed a clear trend towards using permissive licenses like *Apache-2.0* and *MIT/X11* for C#, Java, JavaScript, Python, and Ruby projects, while C and C++ had a trend towards adopting restrictive licenses. Additionally, for Java, we found permissiveness or restrictiveness of a license seems to impact the adoption of newer versions, where permissive licenses are more rapidly adopted. However, this finding was not strongly supported in six languages in this study. Conversely, we found miore support reinforcing that 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.

### 4.2 RQ$_2$: What are the most common licensing change patterns?

Interestingly, we find very different and surprising results in terms of *atomic license changes* in the analysis of C, C++, C#, JavaScript, Python, and Ruby as compared to the results for Java study. In our Java study, we found 204 different *atomic license change* patterns. However, we did not identify any changes in licensing in the other languages. This result was unexpected since we observed that 1,833 projects (out of 16,221 projects) experienced a change in licensing for the Java projects.

In order to understand the lack of licensing, we manually investigated projects written in C. As other studies have found changes in licensing for C based projects,

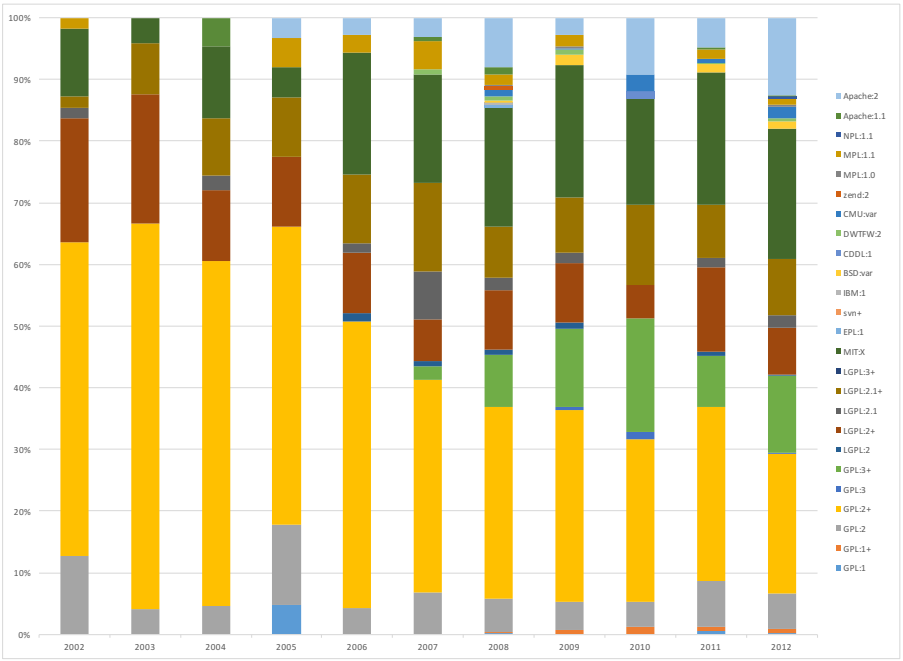


**Fig. 2** Relative License Usage between 2002 and 2012 or C Projects.

these projects seemed likely to help determine the lack of licensing. In our manual investigation, we found that certains projects contained *squashed* histories (i.e., the repositories contained commits that dated back earlier in the development, but the project repository started at a more recent commit and not he earlier commits). Additionally, we observed that projects utilize GitHub to mirror releases and do not contain the actual development of system. Because of this aspect, the projects are uploaded at a mature state and contain licenses already, and this maturity is also likely to indicate more stability with respect to licensing.

In addition, the relatively small number of commits is another factor that may contribute the lack of changes in licensing. Table 3 shows the distributions of the number of commits. We observe that median and third quartile of the number of commits is sufficiently small (less than 100). It is likely that these systems have not undergone enough development where a license change is likely to occur. This observation is consistent with our findings in [32], where we observed that license changes typically occur after a period of more development (measured by the number of commits).

Finally, two other potential factors may impact the lack of license changes: i) limitations arise from the underlying license identification tool (ninka), which may not have been able to accurately capture particular licenses that are more prevalent in other language, and ii) the filtering may remove projects that exhibit delayed licensing, since we aimed to remove lower quality projects. In terms of the former

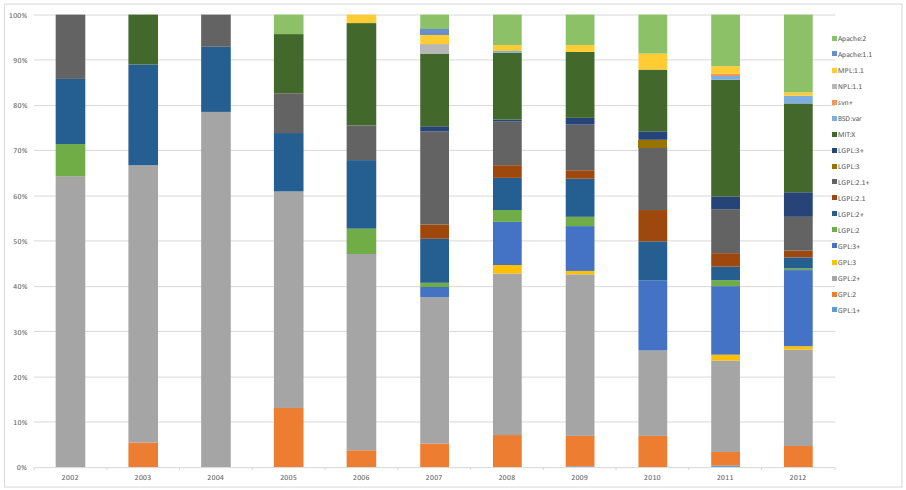


**Fig. 3** Relative License Usage between 2002 and 2012 for C++ Project.

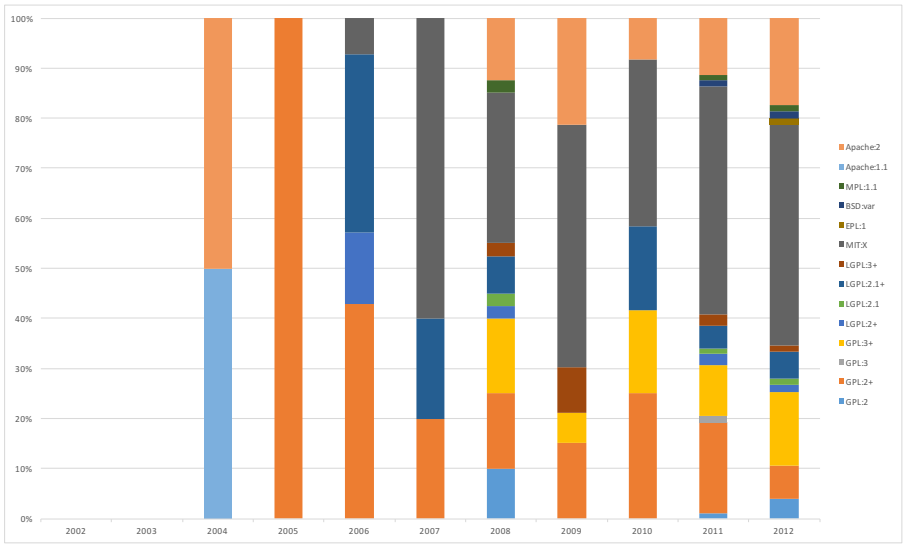


**Fig. 4** Relative License Usage between 2002 and 2012 for C# Projects.

limitation, ninka is the state-of-the-art tool for license identification and has been used in the prior studies in licensing. In the latter case, we aimed to analyze meaningful projects and not class/test repositories, which do not represent real open source development.

While we did not identify license changes in C, C++, C#, JavaScript, Python, and Ruby, our study on Java projects analyzed commits where a license change occurred, with a two-fold goal (i) analyze license change patterns to understand both

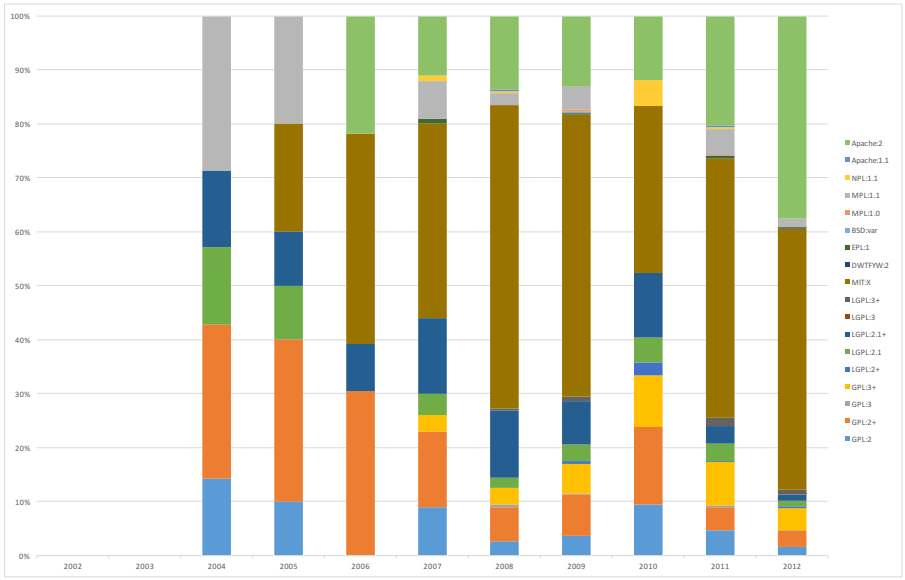


**Fig. 5** Relative License Usage between 2002 and 2012 for JavaScript Projects).

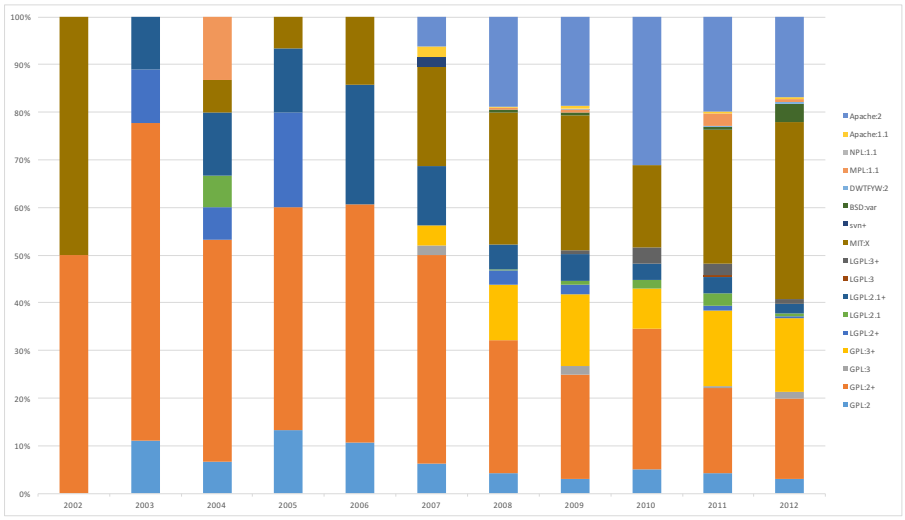


**Fig. 6** Relative License Usage between 2002 and 2012 for Python Projects.

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 4) and dominant local patterns (Table

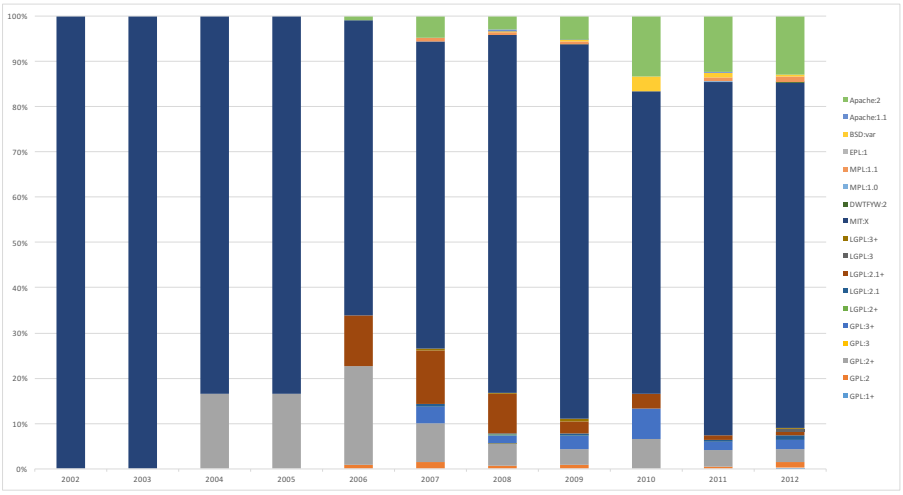


**Fig. 7** Relative License Usage between 2002 and 2012 for Ruby Projects.

5). 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 4 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 4 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 5 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

**Table 3** Distribution of the number of commits for the projects of each language.

| | C | C++ | C# | JavaScript | Python | Ruby |
|---|---|---|---|---|---|---|
| Minimum | 0 | 0 | 0 | 0 | 0 | 0 |
| Quartile 1 | 4 | 4 | 2 | 4 | 2 | 3 |
| Median | 17 | 16 | 9 | 12 | 14 | 11 |
| Mean | 2317 | 516 | 69 | 78 | 122 | 76 |
| Quartile 3 | 86 | 67 | 33 | 38 | 49 | 34 |
| Maximum | 547,518 | 121,570 | 7,241 | 48,939 | 41240.0 | 51,843 |

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 6 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* $\rightarrow$ *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+* $\rightarrow$ *GPL-3.0+* in Table 6 was expected since it was tied for the most prevalent among global *atomic license changes*. Similarly, the patterns of *MIT/X* $\rightarrow$ *Apache-2.0*, *GPL-3.0+* $\rightarrow$ *Apache-2.0*, and *Apache-2.0* $\rightarrow$ *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* $\rightarrow$ *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.

### 4.3 RQ$_3$: To what extent are licensing changes documented in commit messages or issue tracker discussions?

In this study, we did not extract license changes from the six programming languages. Because of this, we would be unable to establish traceability links between the commits in which an *atomic license change* occurred and the issue tracker. Table 7 demonstrates that the keywords require the changed license as well as utilizing commit hashes and dates of both the issues (open date and close date) and commits to generate links between the change and the issue discussions.

**Table 4** 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 5** 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 6** 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 7** 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 |

While we are unable to compare the results of these six languages to the findings from Java as seen in Table 7 and further described later in this section, our GT analysis offers possible rationale for some of the observations from the study on Java projects. For Java, we were unable to create traceability links with closed issues. In sampling crawled issues discussions, we found that issues tracker urls that had been identified by our keyword search no longer existed when we inspected the repository. This observation suggests that some of these issues may have been deleted or that the authors utilize an external issue tracking system.

Table 7 reports the results of the traceability linking between licensing changes and commit notes/issue tracker discussions for Java projects . 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.

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$_3$.** While we were unable to compare our new findings to the study with Java projects, we observed that at least some of the lack of traceability may be due to the removal of issues related to licensing or that developers prefer external issues tracking systems and remove them from GitHub. However, we cannot further compare the 51k+ projects of these six languages to determine whether they also experience that 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 (as seen in Java projects).

### 4.4 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.

Initially, we sampled a total of 913 commits across the three languages: we sampled 226 commits out of 1,133 total commits fo C, 100 commits out of 148 total commits for C#, 139 commits out of 694 total commits for C++, 130 commits out of 650 total commits for Python, 122 commits out of 607 total commits for JavaScript, and 195 commits out of 971 total commits for Ruby. The commits were distributed among the authors and we generated YYY categories from our open coding of the commit messages. We compared these results to our previously sampled and filtered subset of 500 commit messages from the entire set of commit messages among all the Java projects.

The most dominant categories from the commit messages were: *License Addition, Copyright Update, Modififcation To License File,* 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 represents two type of additions of *Added Declared License* and *Generic Addition*. The latter represents a generic commit message of

> *"added a license page to TARDIS."*

The *Added Declared License* type includes commit messages reporting the exact license added. For example, a developer indicated

> *"Add MIT license. Rename README to include rst file extension."*

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. The message only asserts that the particular license governs the system. These results mirror the previous findings, where we observed *Generic Additions*, such as

*"Created LICENSE.md."*

This message is automatically generated when a license is added to an existing GitHub project with GitHub's licensing feature. We also observed similar case where *Add Declared Licenses* appeared in the commit messages.

We also observed commit messaged that indicating *Copyright Updated*. These messages could either specific the particular update, such as updating the copyright year or authors. A particulare example that we observed is:

*"Updated URL license and version number."*

The message indicates minor changes to the license's url and not the actual license terms. Additionally, we observe generic updates where the committer simply indicated that there was an update but no other information or rationale.

Similarly, we observe *Copyright Added* categorized commit messages. These were less prevalent than the updated copyright messaged and typically only indicated that a copyright (or copyright file) had been added. The messages do not offer further meaningful information regarding the nature of the copyright. In some cases, the a particular license may be accompanied, but these messages would be classifed as *License Addition* (due to the fact that a particular license was declared). Both *Copyright Updated* and *Copyright Added* were similarly represented in the commit messages from Java.

As done in the context of Java commit messages, 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

*"Packaging and PEP8 compliance. Suite up"*

indicates that the commit solved some type of *compliance*, but it does not necessarily imply that the commit is related to licensing. In some case, we observe commits that utilize the keywords in a different context. For example, one developer committed,

*"Use setuptools ¿= 2.2 instead of Distribute."*

This message is identified by *distribute*, which is considered a licensing keyword as the licenses specify how sourcce code can be distributed. However, the particular message does not use distribute in terms of code distribution.

Additionally, we observed a new category that we had not previously identifed. The category *Modififcation To License File* repesents reformatting the license file type or file name. For example, developers may change the license file from the default `LICENSE.md` file generated by GitHub to a *.txt* or *.rtf*. Additionally, developers change the file name or file path to the license file. For example, we observed that the parent directory that contained the license file was altered. These cases do not indicate significate changes to the actual licensing, but the way in which the license is represented.

Not surprisingly, *License Change* was not prevalent among the six languages as compared to the Java projects. Since we did not observe any license changes in the dataset, it is expected that the commit messages would seldom (if at all) mention a change to the licensing. However, we did observe commit messages that indicated a license change, which suggests that the underlying approach may have been unable to identify the license of some files causing our analysis to miss some changes to the licensing. For example, we observed the commit message:

> *"Relicensed CZMQ to MPLv2 - fixed all source file headers - removed COPYING/COPYING.LESSER with GPLv3 and LPGv3 + exceptions - added LICENSE with MPLv2 text - removed ztree class which cannot be relicensed - (that should be reintroduced as foreign code wrapped in CZMQ code)."*

The commit message indicate the former licensing, the new licensing, and a change to ensure compliance of the new licensing terms. Interestingly, the particular change demonstrates a move towards a more permissive license, which had been prevalent in our study of Java projects. We previously observed similar commits for those Java projects, where a different license was chosen or the clauses of the license were modified. The following represent each case, respectively:

> *"Switched to a BSD-style license"*

> *"The NetBSD Foundation has granted permission to remove clause 3 and 4 from their software"*

We also observed *License Removal*, *License Dependency Compatibility*, and *Code Removal for Compliance*. While less prevalent, these categeories were also important since it demonstrates that license incompatibilities exist in the projects of these systems and were typically due to dependencies. In order to ensure license compliance, the developers had to remove those dependencies or libraries from the system. Thus, we observe that difficulty with respect to licensing due to dependencies is not limited to Java or C/C++, which has been the predominant focus of the prior studies. Instead, the difficulty is pervasive across languages.

While we were unable to identify *atomic license changes* for C, C++, C#, JavaScript, Python, and Ruby, our analysis of Java *atomic license changes* gave further insights to particular types of license changes. In the analysis, wpecifically 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 for *License Change* category from our previous analysis, such as

> *"Rewrite to get LGPL code."*

> *"Changed license to Apache v2"*

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"*

It indicates the removal of a previously exploited library. Additionally, licenses were removed as developers cleaned up the project.

*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. More importantly, it may suggest that the projects from our dataset written in C, C++, C#, JavaScript, Python, and Ruby also experienced license changes that we were unable to detect.

Additionally, the GT analysis also captured similar support for these categories to a greater extent than the initial study on Java projects. Since we sampled commits from all Java projects, it was infeasible to sample a larger representative number of commit messages. Thus, augmenting the second round benefitted the taxonomy by targetting the commits better. However, we were able to sample statistically representative sample sizes in this work due to pre-filtering the projects. The results corroborate the represenativeness, since we observe the same categories.

Lastly, we investigated the issue tracker discussions in which the issue title suggested it was related to licensing. The analysis of these discussions introduced six new categories: *License Clarification, Reuse, License Compatibility, Choose a License, Missing Licensing,* and *Contributor License Agreement*. *License Clarification* 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 created an issue for a project entitled "What license is this project?" and the developer elaborates in the issue comment explaining the implication of a lack of license,

*"There is no license specified anywhere. I would prefer a non copyleft open source license (no GPL) because I don't thinck I'm going to use it if it's GPL.*

*Something like mpl, apache, new bsd or lgpl. Anyway noone's going to use what you created for anything because at the moment it's proprietary with source code visible. Whatever open source license you would like to use please specify it."*

In the Java projects, we found similar issues 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.

Interestingly, we observed an issue related to *Reuse* where one contributor suggests a dual license to allow for greater reuse in other applications. The contributor stated,

*"Due to incompatibility between GPLv3 and Apache 2.0 it is hard to use python-hpilo from, for instance, OpenStack. It would therefore be helpful if the project code could also be released under a more permissive license, like for instance Apache 2.0 (which is how OpenStack is licensed)"*

The other contributors subsequently utilized the thread to vote and ultimately agree upon the dual license. Not only does this example indicate the consideration for reuse, but it also demonstrates that licensing decisions are determined by all copyright holders and not a single developer.

One interesting observation is that developers also use the issue tracker to track the initial project licensing. We extracted the category *Choosing License*. We observed the open issues "Add LICENSE file," where the issue creator commented,

*"A license needs to be chosen for this repo. All contributors need to agree with the chosen license. A list of contributors is enclosed below"*

SImilarly, we found the same issues from our Java study, such as an issue titled "What license to use" that posed the question of

*"BSD, GNU GPL, APACHE?"*

These observations suggest 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. For example, a non-contributor posted an issue stating,

*"Hi there, at the moment you've declared this project to be wholly MIT-licensed. TyrQuake inherits the software license of the underlying Quake source code, which is GPL-2. Therefore, the Quake and TyrQuake bits in this repository have to be GPL-2, and cannot be MIT. (This is because the GPL-2 includes extra constraints that the MIT does not). Perhaps your intention was that your extra bits are MIT? If so, please clarify that. Additionally, any bits of yours which are derivations from the GPL-2 stuff are in murky water. The simplest thing to do would be to declare the whole repository GPL-2."*

Similarly, we previously observed a 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].

We observed that devleopers also are made aware of missing licenses on the issue tracker as well, generating the *Missing License* categeory. In this case, the project may have files without licensing or files may have not included the appropriate license files may be absent. We observed one developer that indentified the project's " GNU LGPL license is missing." The developer commented,

> *"Under which license is this source code published? This project is heavily based on wiring-pi and rc-switch: rc-switch: GNU Lesser GPL wiring-pi: GNU Lesser GPL The GNU Lesser GPL could be added: http://www.gnu.org/licenses/lgpl.html"*

The developer approaches the missing licensing by asking a question about the licensing, but the comment intends to explain that the LGPL license is missing. 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. We observed a discussion related to textual information regarding a country's designation in the CLA. Similarly, in our previous Java study, 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.

Another important observation that appears to support the supposition from our traceability analysis that developers remove licesning related issues from the issue tracker is that we found links from the time our crawling to our analysis that no longer existed. It is also possible that these cases represent developers that utilize external bug tracking systems as well.

**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.

## 5 Lessons and Implications

The analysis of license usage indicated that licensing practices, especially towards introducing a new license to a project, seem to differ according to the programming language. We were able to reinforce our findings that restrictive licenses tend to survive the release of newer license versions (e.g., *GPL-2.0* survives despite the release of *GPL-3.0*, However, we were unable to support the findings that permissive licenses tend to be replaced by newer versions of the license. In part, the lack of support is due

to the limited diversity in licensing for five out six of the languages, which reinforces that license adoption and usage differs between programming languages. For example, we observed Ruby projects predominantly introduce *MIT/X* to projects, whereas C consisted of *GPL* and *LGPL* licenses. Thus, this work further supports the observations from *Vendome:ICSME15* that tools to assist developers with respect to licensing should also take into account factors like language or domain, since the license usage behaviors is not consistent across languages.

The analysis of the commit messages and issue trackers 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.**

In this work, we were unable to evaluate the traceability of licensing changes, since we did not observe any in our dataset. However, the lack of traceability of licensing changes that was observed in Java is 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* [26] 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/X* license seems to be the most prominent license for this purpose. In fact, we observe a high prevalence of usage of this license in our dataset. 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). We observed issues discussions related to lack of

licensing, since it hindered reuse. The aforementioned suggestion above would also serve to address this problem.

Finally, we observed that some projects may not accurately reflect the behavior of these system well. We observed that certain projects were mirrored mature projects and so the projects were released on GitHub at a mature state without the complete revision history. Additionally, we observed histories that contain earlier commits, but the first recognized commit appears at a later date. Thus, it is possible that a subset of projects did not contain changes to licensing because of this aspect, suggesting these histories we compressed or inaccessible through the git version control system (it is also possible that it is an artifact of migrating the repository from a different version control system).

## 6 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 frequency. 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. However, we present results for six popular programming languages and compare them to our earlier findings with Java. Additionally, our data set is representative of only projects hosted on GitHub and written in Java, C, C++, C#, JavaScript, Python, and Ruby so we do not claim that the results generalize to any project of these languages. 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 3.5) 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.

## 7 Conclusions

We empirically studied phenomena related to license usage and licensing changes in a set of 51,757 projects written in six hosted on GitHub. We compared the findings to our Java study on 16,221 projects. 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;
- While our extension analyzing C, C++, C#, JavaScript, Python, and Ruby did not contain changes in licensing, for the Java study, we observed licensing changes are predominantly toward or between permissive licenses, which ease some kind of derivative work and redistribution, e.g. within commercial products;
- 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;
- The lack of traceability between discussions and related license changes may be in part due to removing licensing-related issues from the issue tracker or utilizing an external issue tracker and removing the issues from GitHub.

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

We would like to thank all the open source developers who took time to participate in our survey. Specifically, we would like to acknowledge developers who provided in-depths answers and responded to follow-up questions. This work is supported in part by NSF CAREER CCF-1253837 grant. Massimiliano Di Penta is partially supported by the Markos project, funded by the European Commission under Contract Number FP7-317743. Any opinions, findings, and conclusions expressed herein are the authors' and do not necessarily reflect those of the sponsors.

## 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, 2010.
26. 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.
27. 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.
28. Oracle. MySQL - FOSS License Exception. http://www.mysql.com/about/legal/licensing/foss-exception/. Last accessed: 2015/01/15.
29. P. Singh 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.
30. T. Tuunanen, J. Koskinen, and T. Kärkkäinen. Automated software license analysis. *Autom. Softw. Eng.*, 16(3-4):455–490, 2009.
31. C. Vendome, M. Linares-Vásquez, G. Bavota, M. Di Penta, D. M. Germán, and D. Poshyvanyk. License usage and changes: A large-scale study of Java projects on GitHub. In *The 23rd IEEE International Conference on Program Comprehension, ICPC 2015, Florence, Italy, May 18-19, 2015*. IEEE, 2015.
32. C. Vendome, M. Linares-Vsquez, G. Bavota, M. D. Penta, D. M. German, and D. Poshyvanyk. When and why developers adopt and change software licenses. In *The 31st IEEE International Conference on Software Maintenance and Evolution, ICSME 2015, Bremen, Germany, September 29 - October 1, 2015*, pages 31–40. IEEE, 2015.
33. C. Vendome, M. L. Vásquez, G. Bavota, M. D. Penta, D. M. Germán, and D. Poshyvanyk. License usage and changes: a large-scale study of java projects on github. In *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension, ICPC 2015, Florence/Firenze, Italy, May 16-24, 2015*, pages 218–228, 2015.
34. Y. Wu, Y. Manabe, T. Kanda, D. M. Germán, and K. Inoue. A method to detect license inconsistencies in large-scale open source projects. In *The 12th Working Conference on Mining Software Repositories MSR 2015, Florence, Italy, May 16-17, 2015*. IEEE, 2015.