# When and Why Developers Adopt and Change Software Licenses

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

***Abstract*—Software licenses legally govern the way in which developers can use, modify, and redistribute a particular system. While previous studies either investigated licensing through mining software repositories or studied licensing through FOSS reuse, we aim at understanding the rationale behind developers' decisions for choosing or changing software licensing by surveying open source developers. In this paper, we analyze when developers consider licensing, the reasons why developers pick a license for their project, and the factors that influence licensing changes. Additionally, we explore the licensing-related problems that developers experienced and expectations they have for licensing support from forges (e.g., GitHub).**

**Our investigation involves, on one hand, the analysis of the commit history of 16,221 Java open source projects to identify the commits where licenses were added or changed. On the other hand, it consisted of a survey—in which 138 developers informed their involvement in licensing-related decisions and 52 provided deeper insights about the rationale behind the actions that they had undertaken. The results indicate that developers adopt licenses early in the project's development and change licensing after some period of development (if at all). We also found that developers have inherent biases with respect to software licensing. Additionally, reuse—whether by a non-contributor or for commercial purposes—is a dominant reason why developers change licenses of their systems. Finally, we discuss potential areas of research that could ameliorate the difficulties that software developers are facing with regard to licensing issues of their software systems.**

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

## I. Introduction

Software licenses are the legal mechanism used to determine how a system can be copied, modified, or redistributed. Software licenses allow a third party to utilize code as long as they adhere to the conditions of the license. In particular, open source licenses are those that comply with the Open Source Definition [4]. Specifically, the goal of these licenses is to facilitate further copying, modifying, and distributing software as long as a set of ten conditions are met (such as free redistribution and availability of source code).

For software to be open source, its creators must choose an open source license. However, there is a large number of open source licenses in use today. They range from highly restrictive (such as the General Public License—*GPL*—family of licenses) to ones with very few restrictions (such as the *MIT* license). The choice of a license will determine if, and how, a given open source software can be reused. This is especially true for libraries that are expected to be integrated and distributed with the software that uses them. Furthermore, the choice of a license might also be affected by the dependencies used (e.g., software that uses a library under the *GPL* requires the software to be *GPL* also, while software that uses a library under the *MIT* license can be under any license, including commercial).

At some point, the creators of open source software must choose a license that: 1) expresses the developers' philosophy; 2) meets their deployment goals, and 3) is consistent with the licenses of the components reused by that software. However, choosing a license is not an easy process. Developers do not necessarily have a clear idea on the exact consequences of licensing (or not licensing) their code under a specific license; for instance, developers ask questions on Question & Answer (Q&A) websites looking for advice on how to redistribute code licensed with a dual license among the other issues (e.g., question 2758409 in Stack Overflow [19] and question 139663 in the StackExchange site for programmers [28]). Also, the problem of license incompatibility between components is not trivial (see [15] for a detailed description of this problem).

During the evolution of a software system, its license might change. In our previous work [30], we empirically showed—for software hosted in GitHub—that license changes are common phenomena. Stemming from the results that we previously captured by analyzing licensing and their changes in software repositories [30], *the goal of this work is to understand when and why changes in licensing happen*. Specifically, this paper reports the results of a survey of 138 developers with the aim of understanding (i) *when* developers consider adding a license to their project, (ii) *why* they choose a specific license for their projects, and (iii) *factors* influencing license changes. The 138 participants are the respondents from a set of 2,398 invitees, i.e., 5.75% of the invitees. We identified such developers by sampling 16,221 Java projects on GitHub, and then subsetting to 1,833 projects where the license changed over time. Of these 138 developers, 52 developers offered insights to the aforementioned questions, while the remaining developers reinforced that licensing decisions are not necessarily made by all contributors, but by a subset that are the copyright holders. The main findings of this study are as the following:

1) Developers frequently license their code early, but the main rationale for delaying licensing is usually to wait until the first release;

TABLE I
PREVIOUS STUDIES INVESTIGATING SOFTWARE LICENSING.

| Study | Purpose | Dataset |
|---|---|---|
| German *et al.* [13] | Investigate the presence of license incompatibilities | 3,874 packages |
| Di Penta *et al.* [10] | Investigate license evolution during a system's maintenance and evolution | 6 systems |
| German *et al.* [15] | Investigate the way in which developers address incompatible licensing | 124 systems |
| German *et al.* [14] | Investigate licensing between copied code fragments in Linux and two BSD distributions | 3 systems |
| Manabe *et al.* [20] | Investigate license change patterns within FOSS systems | 4 systems |
| Sing *et al.* [25] | Investigate the reasons for the adoption of a particular FOSS license | 5,307 projects |
| Sojer *et al.* [27] | Investigate reuse and legal implication of Internet code | 686 developers |
| Sojer *et al.* [26] | Investigate FOSS code reuse | 869 developers |
| Vendome *et al.* [30] | Investigate license usage and changes in FOSS systems and the rationale in the revision history and issue tracker | 16,221 systems |

2) Developers have strong intrinsic beliefs that affect their choice of licenses. Also, open source foundations, such as the Apache Software Foundation, the Free Software Foundation, and the Eclipse Software Foundation exert a powerful influence on the choice of a license;
3) We observed that the change of a license(s) of a system is predominantly influenced by the need to facilitate reuse (mostly in commercial systems);
4) Developers experience difficulties in understanding the licensing terms and dealing with incompatible licenses.

## II. RELATED WORK

Our work is mainly related to (i) the automatic identification and classification of licensing in software artifacts, (ii) empirical studies investigating license adoption and license evolution, (iii) qualitative studies on software licensing. Table I presents prior work in licensing by reporting the main purpose of each study and the corresponding dataset used.

### A. Identifying and Classifying Software Licensing

Automatic identification of software licensing has been widely explored before. To the best of our knowledge, the FOSSology project [17] was the first one aimed at solving the problem of license identification by extracting the licensing information of projects and using machine learning for classification. Another representative project is the ASLA tool by Tuunanen *et al.* [29], which showed an 89% accuracy with respect to classifying the licenses of files in FOSS systems.

The current state-of-the-art automated tool for license identification, Ninka, was proposed by German *et al.* [16]. Ninka relies on pattern-matching in order to identify licensing statements and return the license name and version (e.g., *Apache-2.0*). The evaluation of Ninka indicated a precision of 95%.

Since software is not always distributed *with* or *as* source code, the traditional approaches for license identification that are based on the parsing of the licensing statements are not always applicable (byte-code or binaries do not inherently contain licensing information). To ameliorate this problem, Di Penta *et al.* [9] proposed an approach that uses code search and textual analysis to automatically identify the licensing of jars. The approach automatically queried Google Code Search by extracting information from decompiled code. Additionally, German *et al.* investigated the ability to identify FOSS licensing in conjunction with proprietary licensing by analyzing 523,930 archives [12].

In this paper, we rely on Ninka [16] for license identification, since it is the current state-of-the-art technique. However, our work does not aim to improve upon license identification or classification, but, rather, to understand the rationale behind licensing decisions.

### B. Empirical Studies on Licenses Adoption and Evolution

Di Penta *et al.* [10] investigated license migration during the evolution and maintenance of six FOSS projects. While the authors were unable to find a generalizable pattern among the projects, the results suggested that both version and type of license were modified during the systems' life cycles.

German *et al.* [15] investigated the way in which developers handle license incompatibilities by analyzing 124 FOSS packages and from this investigation they constructed a model that outlines the advantages and disadvantages of certain licenses as well as their applicability. Additionally, German *et al.* [13] conducted an empirical study to (i) understand the extent to which package licensing and source code files were consistent and (ii) evaluate the presence of licensing issues due to the dependencies among the packages. The authors investigated 3,874 packages of the Fedora-12 Linux distribution and they confirmed a subset of licensing issues with the developers at Fedora. Manabe *et al.* [21] analyzed FreeBSD, OpenBSD, Eclipse, and ArgoUML in order to identify changes in licensing. The authors found that each of the four projects exhibited different patterns of changes in licensing.

German *et al.* analyzed fragments of cloned code between the Linux Kernel and both OpenBSD and FreeBSD [14]. They investigated the extent to which terms of the licenses were adhered during the cloning of these code fragments. Similarly, Wu *et al.* [31] found that cloned files have a potential to be inconsistent in terms of licenses (e.g., one has a license, while the other does not). The paper describes the types of inconsistencies and illustrates the problem and the difficulty to resolve it through an empirical study of Debian 7.5.

The most related empirical study to this work is our previous work [30], which analyzed license usage and license changes over 16,221 projects and sought to extract rationale from commit messages and issue tracker discussions. The results indicated a lack of documentation of licensing in both sources. While sharing the same motivation, this work is novel as it investigates *when* and *why* developers choose to license a project or change licensing (as opposed to the extent to which these changes occur) and presents rationale from a survey conducted with actual developers of the projects from our dataset instead of relying just on the rationale from the issue tracker discussions or from commit messages.

TABLE II
DATASET STATISTICS FOR THE 16,211 PROJECTS IN THE STUDY.

| Feature | Total Number |
|---|---|
| Commits | 1,583,708 |
| Files | 941,357 |
| Disk Size of All Repositories (Gb) | 441 |
| Different Licenses | 24 |

### C. Qualitative Studies on Software Licensing

Sing and Phelps [25] studied the reasons behind the adoption of a specific license in a FOSS project. Their results suggest that such a choice is mainly driven by social factors— the adoption of a license in a new project is based on the licenses adopted by socially close existing projects (e.g., projects from the same ecosystem). Their work considered license adoption from a social networking perspective to see how the "licensor" may be influenced toward a particular license(s) based on social proximity. Our work does not investigate latent social connections between developers or the projects in which they contributed. Instead, we directly surveyed the developers to understand their reasoning for adopting a particular license.

Sojer *et al*. conducted a survey with 869 developers regarding reuse of open source code and the legal implications of the resulting code [26]. One key finding was that industry and academic institutions did not prioritize knowledge regarding licensing and reuse. The authors compared a self-assessment to a questionnaire on licensing and found a discrepancy between perceived knowledge and actual understanding of licensing. Additionally, Sojer *et al*. conducted a survey of 686 practitioners regarding reuse of FOSS code and found that licensing of FOSS code was the second largest impedance for reuse [27]. While the authors point to possible reasons for this observation, our study specifically aims to understand the reasons for choosing and changing licenses as well as the types of problems that practitioners face due to licensing.

## III. DESIGN OF THE STUDY

The *goal* of our study is to investigate when developers consider licensing issues and the reasons why developers pick or change licensing in FOSS projects. The *context* consists of software projects, i.e., the change history of 16,221 Java FOSS projects mined from GitHub, and subjects, i.e., 138 practitioners contributing to a subset of the mined projects.

### A. Research Questions

We aim at answering the following research questions:

- **RQ**$_1$ *When and why do developers first assert a licensing to their project?* This research question first examines when developers commit a license to at least one file in FOSS projects hosted on GitHub (i.e., the project goes from no licensing to at least one license). We complement this analysis with questions for developers to understand the actual rationale behind the empirical observations.
- **RQ**$_2$ *When and why do developers change the licensing of their project?* This research question relies on a similar analysis as the previous question, but it specifically investigates licensing changes (i.e., the change from license $A$ to license $B$).
- **RQ**$_3$ *What are the problems that developers face with licensing and what support do they expect from a forge?* This question aims at understanding the problems that developers experience with licensing to better support them. Additionally, we are interested in understanding the expectation that developers may have for support incorporated by forges.

In order to answer our research questions, we consider two perspectives: (i) evidence collected by analyzing projects' change history; and (ii) evidence collected by surveying developers. Both perspectives are explained in the following.

### B. Analysis of the Projects' Change History

To investigate when developers pick or change licensing, we mined the entire commit history of 16,221 public Java projects on GitHub. We first queried GitHub, using the public API [2], to generate project information for all of the publicly available projects. We extracted a comprehensive list of 381,161 Java projects by mining the project information of over twelve million projects and locally cloned all of the Java repositories, which consumes a total of 6.3 Tb of storage space. We randomly sampled 16,221 projects due to the computation time of the underlying infrastructure to analyze the licensing of all the file revisions at commit-level granularity (1,731,828 commits that spanned 4,665,611 files). Table II reports statistics about size attributes of the analyzed dataset and the overall number of different licenses considered in our study.

We relied upon the MARKOS code analyzer [7] to extract the licensing throughout each project's revision history. The code analyzer incorporates the Ninka license classifier [16] in order to identify the licensing statements and classify the license by family and version (when applicable) for each file. The code analyzer mined the change log of the 16,221 projects and extracted commit hash, date, author, file, commit message, change to file (**A**ddition, **M**odification, or **D**eletion), license change (Boolean value), license name and version (reported as a list, when multiple licenses are detected).

The data extraction step for the 16,221 projects took almost 40 days, and a total of 1,731,828 commits spanning 4,665,611 files were analyzed. In the case of *BSD* and *CMU* licenses we only reported a variant of either case, since Ninka was unable to identify the particular version. In the case of *GPL* and *LGPL*, it is possible for the license to have an exception that allows developers to pick future versions of that license and we annotate the license with a "+" (e.g., *GPL-2.0+* signifies that the terms of *GPL-3.0* can also be used).

To identify licensing changes, we followed the same procedure exploited in our previous work [30]. In particular, we identify a commit $c_i$ as responsible for introducing a license in a code file $F$ if before $c_i$ Ninka did not identify any license in $F$, while after $c_i$ a license in $F$ is retrieved (i.e., *No License* $\rightarrow$ *Some License* transition on $F$). Instead, we consider $c_i$ as a licensing change if the license type and/or version detected by Ninka on $F$ before $c_i$ is different than the one detected after $c_i$ (i.e., *Some License* $\rightarrow$ *Some Other License* transitions).

### C. Analysis of the Developers' Survey

To investigate the reasons why developers add/change the license(s) of their systems, we surveyed the developers who made licensing changes in the systems to which they contributed. To find potential developers for our survey, we utilized the results of our quantitative analysis. From the 16,221 projects that we analyzed, we found 1,833 projects that had experienced either a delayed initial license addition (i.e., *No License* $\rightarrow$ *Some License* transition happened after the first project commit) or licensing change (i.e., *Some License* $\rightarrow$ *Some Other License*) over their change history. We included both scenarios to understand the rationale behind both $\mathbf{RQ}_1$ and $\mathbf{RQ}_2$, which required a change in licensing. For each of these projects, we used the version control history to extract the set of all its contributors. From the 1,833 projects with licensing changes, we identified a total of 2,398 valid developers e-mail address, whom we targeted as potential participants for our study. By valid, we refer filtering out contributor e-mail addresses matching the following two patterns — "[user]@locahost.*" or "[user]@none.*"— since they pointed to clearly invalid domains. We also removed developers of the Android framework, since these have always been licensed under the *Apache* license. The 2,398 developers were invited via e-mail to fill-in an online survey hosted on Qualtrics [5] (the survey answers were all anonymous). This e-mail invitation included (i) a link to the survey, and (ii) a description of the specific licensing addition/change(s) we observed in their project's history. After being contacted, some developers offered further insights regarding these changes by directly responding to our email. In total, we emailed 2,398 individuals and received 138 responses to the survey and 15 follow-up emails in which developers volunteered additional information. Overall, we had a response rate of 5.75% of the developers we contacted.

The survey consisted of seven questions (Q1-Q7); Q7 was optional (only 12 participants answered it). Tables III and IV list the survey questions and the responses of the developers. Q1 and Q2 were dichotomous questions. These questions were used to ensure that the respondents were involved in determining the project's licensing. If a respondent did not answer "yes" to Q2, the survey ended for the participant. Out of 138 participants, 62 responded "no" to the Q2 and so they were ineligible for the remaining questions (Q3-Q7). Questions Q3 to Q6 were multiple choice questions and included the "Other" option. If the respondents chose "Other", they could further elaborate using an open-ended field. Question Q7 was optional and open-ended. We chose to make it optional, because some developers may not agree that the forge should be responsible for features supporting licensing. Out of 138 respondents, 76 developers were eligible for the entire survey (Q1-Q7) as per their response to Q2, but only 52 of those individuals completed the survey.

Since questions Q3-Q7 also included open-ended responses, we relied on a formal grounded-theory [8] coding of the open-ended responses. Three authors read all the responses and categorized each response that represented the developer's rationale. The categories from the three authors were analyzed and merged during a second round to obtain a final taxonomy of categories. The Tables in Section IV present the final results of the grounded-theory process.

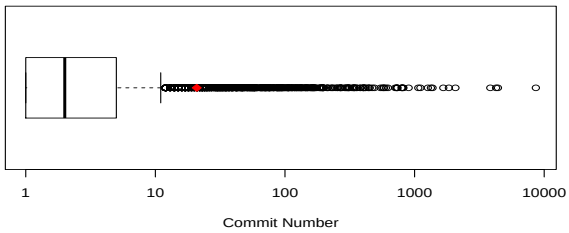


Fig. 1. Distribution of commits with license adoptions (log scale).

## IV. Results

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

### A. When are the licenses added to FOSS projects?

Fig. 1 shows the distribution of the number of commits in which licenses were introduced into the projects within our dataset (e.g., a license introduced during the tenth commit will be represented by the number 10). We present the raw commits in log scale due to outliers from large commit histories. At least 25% (first quartile) of the projects were licensed in the first commit (Fig. 1). The median was also at two commits and third quartile was at five commits. This observation indicates that FOSS projects are licensed very early in the change history with over 75% of the projects having a license by the fifth commit. Assuming (but this might not be always the case) that the observed history corresponds to the entire project history, this result suggests that licensing is important to developers. It is interesting to note that the mean commit number for adding a license is 21 and the maximum value is 623 commits. These two values are indicators of a long tail with a small number of projects that consider licensing late in the change history.

**Summary for $\mathbf{RQ}_1$ (Project History Results):** we observed that developers consider licensing early in the change histories of FOSS projects. While there are projects that assert a license after a larger number of commits, 75% of our dataset had a license asserted within the first five commits. Thus, the data suggests that most of the projects adopt licenses among the very first commit activities.

### B. Why are licenses added to FOSS projects?

Table III reports the responses to Question 3 (Q3) of our survey in which we tried to ascertain the rationale behind the initial project licensing. 30.8% of developers indicated that the community influences the initial licensing. One explanation for the high prevalence of this response is that certain FOSS communities stipulate and enforce that a particular license must be used. For example, the Apache Software Foundation requires that its projects or the code contributed to their projects are licensed under the *Apache-2.0* license. Instead,

the *Free Software Foundation* promotes the use of the *GPL* and *LGPL* family of licenses.

19.2% of developers chose the license with the goal of making their project reusable in commercial applications. These responses also indicate a bias toward more permissive licenses that facilitate such usage while restrictive licenses can discourage such usage, since they require that a system is licensed under the same terms. This finding provides a partial explanation for the trend toward more permissive licenses we observed in our previous work [30].

The results of our survey also show that licensing-related decisions are impacted by inherent developer bias. 15.4% of developers supplied answers that we categorized as moral-ethical-beliefs. An example of this category was the response by one developer indicating, "*I always use GPL-3.0 for philosophical reasons*." Similarly, a different developer echoed this comment stating "*I always licence GPL, moral reasons.*"

Satisfying a dependency constraint (i.e., the need to use a license based on the license of dependencies) was a relevant reason (9.6% - 7.7% picking the explicit option and 1.9% with an "Other" response categorized as dependency constraint). This result is important, since little work has been done to analyze licensing across software dependencies. This problem also poses challenges in both identifying all of the necessary dependencies as well as the license(s) of those dependencies. Some automated build frameworks like *Maven* [6] or *Gradle* [3] attempt to ameliorate this difficulty by listing dependencies in a file that drives the building process (e.g., Project Object Model file in Maven). However, licensing is not a required field in those files.

The remaining answers to this question described situations in which the license was inherited by the initial founders and persisted over time. Also, the companies have policies to specifically dictate a licensing convention. In the latter case, the respondent indicated that "*company (...) policy is Apache-2.0*" (company name omitted for privacy). It was also interesting to see that nobody choose a license based on requests by outsiders.

Lastly, we identified a category in licensing changes that related to license adoption and not changes. 7.7% of developers respond to our question on licensing changes that indicated the license was missing and it was added in a later commit. For this case, we added (License Addition) to the category for Q4 in Table III. The developers noted that "*Setting the license was just forgotten in the first place*" and "*Accidentally didn't include explicit licence in initial commit*". These cases are also important, since it can create inconsistencies within the system or mislead non-contributors that the project is unlicensed or licensed under incompatible terms. This result further reinforces that developers view early license adoption as important, but the lack of a license may be a mistake.

**Summary for $RQ_1$ (Survey Results):** the initial licensing is predominantly influenced by the community to which a developer is contributing. Subsequently, commercial reuse is a common factor, which may reinforce the prevalence of permissive license usage. While reuse is a consideration, non-contributors do not seem to impact the initial licensing choice. We also found that the inclusion of a particular dependency can impact the initial licensing of a project.

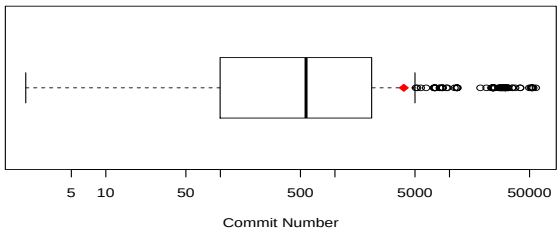


Fig. 2. Distribution of commits with license changes (log scale).

### C. When are licenses changed in FOSS projects?

Fig. 2 shows the distribution of when licenses were changed in the projects within our dataset (i.e., *Some license→Some Other License*). As in the previous section, we present the raw commit number in which the changes occurred (log scale due to outliers from large commit histories). Interestingly, the minimum value was the second commit (i.e., a license changed right after its addition in the first commit). More generally, 25% of license changes occur in the first 100 commits. The median value is 559 commits while the mean is 3,993 commits. The third quartile (2,086 commits), quite smaller than the mean, suggests a long tail of license changes occurring late in the projects' change histories. The maximum commit number with a license change was commit 56,746. Numbers at this extreme would cause the larger mean value compared to the median. Overall, the data suggests that certain projects change licenses early in the change history; however, the license changes are much more prevalent in later commits.

**Summary for $RQ_2$ (Project History Results):** we observed that developers change licensing later in the change history of the FOSS projects. While there are projects that change licensing early, our first quartile was 100 commits and third quartile was 2,086 commits, demonstrating more substantial development occurred before changing licensing.

### D. Why are licenses changed in FOSS projects?

Table III shows the responses to Question 4 (Q4) of our survey in which we investigated the rationale behind license changes. Allowing reuse in commercial software was the most common reason behind licensing changes (32.7%). This option was also the second most prevalent for choosing the initial license (19.2% of developers). Combining these two results, it is clear that the current license of a project is heavily affected by its need to be reused commercially. As previously stated, this result qualitatively supports the observation from our previous work [30], where we observed that projects tend to migrate toward less-restrictive licenses.

7.7% of developers changed licensing due to community influence. This response was a more significant factor for the initial choice in licensing, but it further emphasizes the impact that a community can assert. One developer commented, "*community influence (contributing to Apache's projects)*". Similarly, two developers commented about the influence the

Eclipse Foundation exercised over license changes in their projects. Interestingly, one developer reported: "*I wanted to use the most common one for OSS Java projects*". This response suggests that a particular license may pick up more momentum and spread for a particular language. Interestingly, we observed that 7.7% of the developers were willing to change the licensing due to requests from non-contributors. The fact that this response was more prevalent for changing licensing than choosing the initial license may be influenced by outsiders waiting until a project is stable or mature before inquiring about particular licensing.

We also observed that both the change in license(s) of a dependency or using a new dependency prompted developers to change licenses (5.8% of developers for both cases). This observation further demonstrates the difficulty or impact that dependency can have with respect to licensing. It also suggests that there could be inconsistencies between the licensing of a system and its dependencies.

Moral-ethical-beliefs are also a reason for 5.8% of developers. Interestingly, we observed both the beliefs of developers and beliefs of a philantropist, who is funding the project's development. While one developer acknowledged, "*I simply wanted to pick a 'free' license and chose Apache without much consideration,*" another developer indicated that "*Philanthropic funders encouraged us to move to GPL3, as well as our own internal reflection on this as we came to understand GPL3 better.*" In the former example, it is notable that the developer's concern was not the impact of the *Apache* license in particular, but primary motivator was any *free* license (i.e., FOSS license). The latter indicates that the individuals funding the projects can influence the licensing. While the developers were not coerced to change to the *GPL-3.0*, they were still influenced by the beliefs of the individuals funding the system's change history.

**Summary for $RQ_2$ (Survey Results):** developers seem to change licensing to support reuse in commercial systems. While community influence still impacts changing licensing, it appears to be a less significant factor with respect to license adoption. Based on our survey results, the reasons behind changing licensing are more diverse and more evenly distributed among the topics than we observed in the selection of the initial license.

### *E. What are the problems that developers face with licensing and what support do they expect from a forge?*

Table IV shows the results for Questions 5-7 (Q5-Q7) that investigate both the problems that developers experience with licensing and expected licensing support from the forge.

In Q5, we investigated the problems related to licensing that developers have experienced. 23 out of 52 developers (44.2%), explicitly mentioned "No problem" in the "Other" field. For those who recognized problems, the main reason was the inability of others to use the project due to its license (17.3%). Since developers consider this a problem, it suggests that developers are interested in allowing broad access to their work. However, they may be constrained due to desired protections (e.g., patent protect from *Apache-2.0* or *GPL-3.0*) or external factors, like the licensing of dependencies (external since the developers cannot change those licenses).

Additionally, developers indicated that choosing the correct license was difficult for them (13.5%). The litigious nature of these licenses can lead to misinterpretations by developers. For example, the Apache Foundation states on their webpage that "*The Apache Software Foundation is still trying to determine if this version of the Apache License is compatible with the GPL*" [1]. Additionally, 5.8% developers indicated that they experienced misunderstandings with respect to license compatibility. To make matters worse, 9.6% of the developers experienced compatibility problems with dependencies. Therefore, developers not only faced difficulty while determining the appropriate license, but they also misunderstood the compatibility among licenses and experienced incompatibility between their project's licensing and a desired dependency's licensing.

Developers also experienced difficulties with their users misinterpreting or not understanding the terms of their license. One developer stated that "*Users do not read/understand the license, even though it is a most simple one.*" This result poses two possible problems — either users (i.e., developers looking to reuse the code) ignore the actual licensing text or they struggle to interpret even the easier licenses. The former would demonstrate a bigger problem in that users do not take licensing seriously, while the latter demonstrates that the difficulty in understanding licensing is more extensive than just very litigious licenses. Reinforcing the second scenario, another developer noted the problem was "*Just the usual challenges of talking with potential commercial partners who do not understand the GPL at all*". By phrasing the comment with *the usual challenges*, it suggests that the developer had repeated experience with partners unable to understand licensing. This is not necessarily an isolated case, but rather potentially widespread experience shared by other developers.

Regarding the support provided by the forge, in this case GitHub, we investigated the impact of a feature added to help document the license of a project—see Q6 in Table IV. This feature was added as a response to the criticism from some practitioners [24]. While 36.5% of developers did not have access to the feature at the time they created their project, the interesting result is that more than half (51.9%) of developers were not influenced by the availability of such a tool. Additionally, the "Other" responses indicated that the feature would not have had an impact on their choice (3.8%) and a single developer specifically chose not to license her project, leading to a combined 58% of developers that were unaffected by this feature. Thus, our data suggests that this GitHub feature did not affect/influence developers when licensing (or not) software hosted in GitHub.

Finally, we received 11 responses to our optional question (Q7) concerning whether forges should provide features that assist the licensing of their software. Since GitHub has been criticized by practitioners [24] for a lack of licensing consideration, this question seeks to understand the features that practitioners expect from a forge to this end. 10 out of 11

TABLE III
SURVEY RESULTS. QUESTIONS 1 TO 4: DEVELOPER INVOLVEMENT AND RATIONALE FOR CHOOSING OR CHANGING LICENSES.

| Question/Answer | #D | % |
|---|---|---|
| **Q1. Were you involved in changes occurring to parts of the system that underwent license changes?** | **138** | |
| Yes | 75 | 54.3% |
| No | 63 | 45.7% |
| **Q2. Were you involved in determining the license or change in license of the project or some of its files?** | **138** | |
| Yes | 76 | 53.7% |
| No | 62 | 46.3% |
| **Q3. How did you determine/pick the initial license for your project or files in your project?** | **52** | |
| Dependency constraint | 4 | 7.7% |
| Community influence (e.g., contributing to Apache projects) | 16 | 30.8% |
| Requests by non-contributors to reuse your code | 0 | 0% |
| Interest of reuse for commercial purposes | 10 | 19.2% |
| Other (please specify) | 22 | 42.3% |
| — Closed-source | 1 | 1.9% |
| — Company-policy | 2 | 3.8% |
| — Dependency-constraint | 1 | 1.9% |
| — Inherit-license | 3 | 5.8% |
| — Moral-ethical-belief | 8 | 15.4% |
| — Project-Specific | 2 | 3.8% |
| — Social-trend | 2 | 3.8% |
| — None | 3 | 5.8% |
| **Q4. What motivated or caused the change in license?** | **52** | |
| License of dependencies changed | 3 | 5.8% |
| Using a new library imposing specific licensing constraints | 3 | 5.8% |
| Allow reuse in commercial software | 17 | 32.7% |
| Requests by non-contributors to reuse your code | 4 | 7.7% |
| Other (please specify) | 25 | 48.1% |
| — Change-to-license-text | 2 | 3.8% |
| — Community-influence | 4 | 7.7% |
| — Fix-incorrect-licenses | 1 | 1.9% |
| — Improve-clarity | 1 | 1.9% |
| — Missing-license (License Adoption) | 4 | 7.7% |
| — Moral-Ethical-belief | 3 | 5.8% |
| — More-permissive-license | 1 | 1.9% |
| — New-license-version | 2 | 3.8% |
| — Personal-Preference/Project-specific | 1 | 1.9% |
| — Private-to-public-project | 1 | 1.9% |
| — Promote-Reuse | 1 | 1.9% |
| — Unclear | 1 | 1.9% |
| — None | 3 | 5.8% |

TABLE IV
SURVEY RESULTS. QUESTIONS 5 TO 7: PROBLEMS WITH LICENSING AND EXPECTED SUPPORT FROM A FORGE.

| Question/Answer | #D | % |
|---|---|---|
| **Q5. What problems (if any) have you experienced due to license selection in terms of code reuse?** | **52** | |
| My license was not compatible with desired dependencies | 5 | 9.6% |
| Others were unable to use my project unless I re-licensed it | 9 | 17.3% |
| A dependency changed licenses ad was no longer compatible | 1 | 1.9% |
| There was a misunderstanding of compatibility between licensing terms of two licenses | 3 | 5.8% |
| Choosing the correct license was difficult/confusing | 7 | 13.5% |
| Other (please specify) | 27 | 52.9% |
| — Code-unavailability | 1 | 1.9% |
| — Lack-of-undersanding-by-Users | 2 | 3.8% |
| — Unique-New-License | 1 | 1.9% |
| — No problems | 23 | 44.2% |
| **Q6. Did GitHub's mechanism for licensing impact your decision on licensing your project?** | **52** | |
| Yes, it caused me to license my project | 3 | 5.8% |
| No, I already planned on licensing | 27 | 51.9% |
| No, I did not want to license at project creation | 1 | 1.9% |
| Such a mechanism was not yet available when I created my project | 19 | 36.5% |
| Other (please specify) | 2 | 3.8% |
| — No impact | 2 | 3.8% |
| **Q7. What kind of support would you expect from the forge/GitHub to help you managing licenses and licensing compatibility issues in your software?** | **11** | |
| None | 10 | 90.9% |
| License Checker and License Selection Wizard | 1 | 9.1% |

participants answered "*None*". Of those 10 developers, only one explained that a third party tool should handle license compatibility analysis. The respondent indicated that the ideal tool would utilize the various forges and build frameworks to be a dependency graph of license compatibility stating the following:

*"This is the job of a 3rd party tool IMO since neither github nor forge do or should own all open source deps. A 3rd party tool ideally would know about github, bitbucket, etc + poms and pom license fields, etc and form a comprehensive dep-graph license compat view given a node."*

Another developer noted, "*None. From our perspective it really isn't that hard to put copyright and licence notices in our source files.*" This comment is interesting since it conflicts with results from Q4, where developers indicated that licenses were sometimes missing or an incorrect license was used.

The only developer wishing support from the forge indicated a desire for a license compatibility checker and a license selection wizard. This developer commented the desire for two particular features stating the following:

*"1) License compatibility checker - verify the license of your project with the license of included support software (gems, libraries, includes) and alert user to potential conflicts. This could also be used for the use case that you want to adopt a piece of software to add to an existing project - is it compatible? 2) License selection wizard - when you begin a project, the wizard can ask you a series of questions (do you want to allow commercial use, do you require mods*

*to be licensed the same as original, etc) and then suggest a license for the project."*

While only one developer wanted support from the forge, this single developer's comments seem to address many of the problems and difficulty with respect to licensing for which we found evidence in Q6 of the survey.

**Summary for $RQ_3$ (Survey Results):** although 44.2% of the developers surveyed indicated that they have not experienced problems with licensing, the remaining respondents provided a diverse set of answers. They primarily were related to license incompatibility or difficulty understanding the licensing. Lastly, the survey indicated that GitHub's mechanism to encourage or aid in licensing was not necessary or unavailable to the surveyed developers. We also found that most developers did not expect support from the forge, but one did indicate the desire for a third-party tool. However, one developer did express interest in forge's support, and the comments aligned with our results regarding problems that developers actually faced.

## V. Lessons and Implications

**Intrinsic beliefs of the developers.** The first important observation is that the participants have a bias toward FOSS licensing from an ethical perspective. 52% of the respondents indicated (Q6) that they planned on licensing the project prior to creation; only 6% of the respondents (Q6) were influenced to license their project due to GitHub's licensing feature (i.e., a combo list with license names). Similarly, the "Other" responses regarding the reason for a project's initial licensing (Q3) indicated a sense of obligation. For example, one developer said: "*It was the only moral and ethical choice*".

**Delayed licensing.** Developers do not necessarily have to decide to open source from the beginning and delay doing it. While we empirically observed early license adoption in general, one developer wrote in an email that they waited to choose a license: "*this project just didn't have a license on day 1 and it was added at first release.*" Similarly, one developer responded to the survey that licensing changed due to "*change private to public project*". This observation suggests that licensing is still important to these developers, but it may not be considered relevant until the project reaches a certain level of maturity. Thus, there is the need for tools to add and verify licensing information of a system at any given point in time.

**Community and organizational influence.** Our results indicate that communities, and in particular FOSS foundations (such as the Apache Software, Eclipse, and the Free Software foundations) exert powerful influence on the choice of a license by its developers. About 31% of the participants responded that initial licensing is done by following community's specific licensing guidelines. Improving or developing on top of existing software from a foundation mostly requires using the same license aligning with foundation's philosophy.

**License misunderstanding.** The survey stresses the need for aid in explaining licenses and the implications of their use. About 20% of the respondents highlighted that licensing is confusing and/or hard to understand (Q5): 13.5% of respondents indicated that developers—both the authors and the users—find licensing confusing or difficult (Q5), and 6% of developers also noted that there were misunderstandings between license compatibility. Additionally, one "Other" respondent stated, "*Users do not read/understand the license, even though it is a most simple one*," which suggests that developers experienced misunderstanding whether on their own or by users.

**Reuse for commercial distribution.** The results regarding licensing changes indicated that commercial usage of code is a concern in the open source community. We found that practitioners used permissive licenses to facilitate commercial distributions, in some cases they change to a more permissive license for this purpose.

**Dependency influence.** A software system must choose its dependencies so to avoid conflicts due to incompatibilities between the system's license(s) and the depending components' license(s). Similarly, others will choose to use a particular software system based on its license. Thus, the change of a license in a system has the potential of creating a chain reaction: those that use it might need to change their license, or drop it as a dependency; for the system changing license, the potential pool of reusable components will change accordingly—it might need to drop a different dependency, or it might be able to add a dependency with a previously incompatible license.

**Forge's support.** Most of our respondents do not expect any licensing support from the forge. It is likely that the individuals that benefit the most from licensing support in the forge are those who are looking to reuse software. This is supported by our results that indicate that the license(s) of dependencies is an important consideration, since it might impact the ability to reuse the dependency or require a change of the license(s) of the software that uses it. Thus, compliance-oriented features may aid developers to ensure they can legally reuse software.

Finally, our results demonstrate that external factors like community, license prevalence, and licenses of dependencies have an important impact on licensing.

A feature provided by the forge to support domain suggested licensing could benefit practitioners. Since developers indicated that licensing is difficult, a more informative feature could help practitioners determine the appropriate licensing. For instance, the current licensing support feature provided by GitHub feature is not particularly informative for developers. Basically, it provides a link to choosealicense.com, but does not provide further guidance to the developer. Also, it does not cover issues related to compatibilities at all. Moreover, applications within the same domain may be utilizing some of the same dependencies or require similar grants for redistribution and reuse. To better support developers, a forge could include a domain analysis feature to detect similar applications [22] and suggest to the developer/maintainer the license used by similar systems (if no other criteria has been considered, such as community or dependencies).

## VI. Threats to Validity

Threats to *construct validity* relate to the relationship between theory and observation, and can be mainly due to imprecision extracting licensing and results from the developer survey. In order to identify the licenses, we relied on Ninka [16], which has been empirically evaluated indicating a precision of 95%, when it is able to identify the license (85% of the time in the same study showing the precision). In order to classify the free responses, we conducted a formal Grounded Theory analysis with two-author agreement. In particular, all of the responses were read and categorized by three authors and the agreement of two of them was considered necessary. Another threat concerns the fact that, possibly, GitHub could have mirrored only a fraction of the projects' change history; hence, it is possible that the first commits in GitHub may not correspond to the first commits in the projects' history. Finally, the response rate of our study is 5.75%, below the response rate often achieved in survey studies [18], i.e. 10%. However, explicitly targeting original developers is usually challenging because many of them may not be active, the email addresses are invalid, or even impossible to contact because they are no longer using the email addresses we collected.

Threats to *internal validity* relate to internal, confounding factors that would bias the results of our study. In analyzing both license introduction and licensing changes, we considered the commit in which we observed the phenomena as an instance to ensure we did not introduce duplicates. We only excluded developers of projects from the Android framework since the project has always been *Apache* licensed. Therefore, we did not have a bias while selecting developers. To address lack of coverage of our original options in our survey, we added a free form option "Other" to each question. In addition, we only presented the full survey to developers that indicated that they were involved in the licensing decision(s). Another possible threat to internal validity concerns the fact that, possibly, the 138 respondents decided to participate to the survey because they had greater interest in licensing problems than others. However, results shown in Section IV suggest that this is not the case, e.g., respondents comprise people who are directly involved in the licensing, but they did not necessarily experience any licensing problems.

Threats to *external validity* relate to the ability to generalize the results from the study, and we do not assert that these observations are representative of the FOSS community. While we randomly sampled the projects from GitHub, we only did it for Java projects. Thus, other languages and forges may demonstrate different behavior as well as the developers of those projects may have different beliefs. However, GitHub is the most popular forge with a large number of public repositories. A larger evaluation on multiple forges and projects in other languages is necessary to understand when licenses are adopted and changed in the general case. Additionally, we surveyed actual developers of these projects. While we do not claim that the rationale is complete, the conclusions represent explicit feedback as opposed to inferred understanding. Therefore, the rationale is a definitive subset. We do not claim that these results apply in the context of closed source systems, since we required source code to identify licensing.

Finally, to limit this threat to external validity, we examined the diversity of our data set using the metrics proposed by Nagappan *et al.* [23]. To understand the diversity, we matched the projects in our dataset against the projects mined by Boa [11], finding 1,556 project names that matched between the two datasets. We used these 1,556 projects to calculate our diversity score across six dimensions. The results were 0.45 for programming language, 0.99 for developers, 1.00 for project age, 0.99 for number of committers, 0.96 for number of revisions, 0.99 for number of program languages, suggesting that our dataset is diverse excluding the programming language score (impacted by selecting Java projects). Overall, our score was 0.35, which suggests that we cover over a third of FOSS projects with 9.5% of our dataset.

## VII. Conclusions

We investigated the reasons on *when* and *why* developers adopt and change licenses during evolution of FOSS Java projects on GitHub. To this aim, we conducted a survey with developers that contributed changes to the projects that included licensing changes. We observed that developers typically adopt a license within the first few commits, suggesting that developers consider licensing as an important task. Similarly, we observe that most licensing changes appear after a non-negligible period of development as visible from the observed history. We then explored the reasons for the initial licensing, license changes, and problems experienced by developers with respect to software licensing. We observed that developers view licensing as an important yet non-trivial feature for their projects. License implications or compatibility are not always clear and so they can lead to changes. Additionally, there are external factors influencing the projects' licensing, such as community, purpose of usage (i.e., commercial systems), and use of third-party libraries. While developers did not strongly indicate an expectation for licensing support by the forge, it is evident that third-party tools or features within the forge would aid developers in helping to deal with licensing decisions and changes.

## 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] Apache License, Version 2.0 (current) https://www.apache.org/licenses/. Last accessed: 2015/03/23.

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

[3] Gradle.https://gradle.org/.

[4] Open Source Definition http://opensource.org/osd.

[5] Qualtrics http://www.qualtrics.com/.

[6] Apache. Apache maven project. https://maven.apache.org/.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] D. M. Germán and M. Di Penta. A method for open source license compliance of java applications. *IEEE Software*, 29(3):58–63, 2012.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] R. M. Groves. *Survey Methodology, 2nd edition*. Wiley, 2009.

[19] J. Hartsock. jquery, jquery ui, and dual licensed plugins (dual licensing) [closed] http://stackoverflow.com/questions/2758409/jquery-jquery-ui-and-dual-licensed-plugins-dual-licensing. Last accessed: 2015/02/15.

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

[21] 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.

[22] C. McMillan, M. Grechanik, and D. Poshyvanyk. Detecting similar software applications. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 364–374, Piscataway, NJ, USA, 2012. IEEE Press.

[23] 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.

[24] S. Phipps. Github needs to take open source seriously http://www.infoworld.com/d/open-source-software/github-needs-take-open-source-seriously-208046.

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

[26] M. Sojer and J. Henkel. Code reuse in open source software development: Quantitative evidence, drivers, and impediments. *J. AIS*, 11(12), 2010.

[27] M. Sojer and J. Henkel. Code reuse in open source software development: Quantitative evidence, drivers, and impediments. *Journal of the Association for Information Systems*, 11(12):868–901, 2010.

[28] J. T. Confusion about dual license (mit/gpl) javascript for use on my website http://programmers.stackexchange.com/questions/139663/confusion-about-dual-license-mit-gpl-javascript-for-use-on-my-website. Last accessed: 2015/02/15.

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

[30] 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.

[31] 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.