2017 IEEE 25th International Conference on Program Comprehension (ICPC)

Do Software Developers Understand Open Source Licenses?

Daniel A. Almeida and Gail C. Murphy
Department of Computer Science
UBC, Vancouver Canada
{daa,murphy}@cs.ubc.ca

Greg Wilson
Software Carpentry Foundation
Toronto Canada
gvwilson@software-carpentry.org

Mike Hoye
Mozilla Corporation
Toronto Canada
mhoye@mozilla.com

***Abstract*—Software provided under open source licenses is widely used, from forming high-profile stand-alone applications (e.g., Mozilla Firefox) to being embedded in commercial offerings (e.g., network routers). Despite the high frequency of use of open source licenses, there has been little work about whether software developers understand the open source licenses they use. To our knowledge, only one survey has been conducted, which focused on which licenses developers choose and when they encounter problems with licensing open source software. To help fill the gap of whether or not developers understand the open source licenses they use, we conducted a survey that posed development scenarios involving three popular open source licenses (GNU GPL 3.0, GNU LGPL 3.0 and MPL 2.0) both alone and in combination. The 375 respondents to the survey, who were largely developers, gave answers consistent with those of a legal expert's opinion in 62% of 42 cases. Although developers clearly understood cases involving one license, they struggled when multiple licenses were involved. An analysis of the quantitative and qualitative results of the study indicate a need for tool support to help guide developers in understanding this critical information attached to software components.**

***Keywords*-open source, software licenses, survey**

I. Introduction

Software developers increasingly use open source software to build applications. As one example, Sonatype, the organization behind the Central Repository that helps Java developers access open source components as part of the build of an application, reports that the average number of open source components relied upon in 2014 was 106 per application [1].

Most often when an open source software component is selected and used in a new application, it is accessed via an application programming interface (API). Over the last twenty years, there has been substantial investigation of challenges developers face in understanding and using APIs (e.g., [2], [3], [4]). This previous work has focused primarily on technical aspects of components, such as the code and related documentation. However, when developers who are working on a software project, whether open or closed source, choose to use an open source component, they must also understand and determine if the software to be used is licensed in a way that is compatible with their intended use of the component.

If there were only one or two open source licenses in existence, understanding the licenses and how they can be used would be reasonably straightforward. Unfortunately for developers, there are many open source licenses. As just one example of the diversity of licenses for software in one programming language, Vendome and colleagues found over 25 licenses used in a sample of Java GitHub projects [5]. When a developer makes use of open source software, the use may take many forms, such as through copying a code snippet, using a self-contained library, extending code that is structured as a framework, to name just a few. Not all of the ways in which a developer may wish to make use of open source code may be allowed by the license applied to that code and the way in which the code is used may affect the resultant license of the application being built. The intricacies of licenses and how they apply in different situations can result in license incompatibility issues. Germán and colleagues report finding license incompatibility issues as a result of dependencies between software with different licenses [6]. Hermel and colleagues describe how the `gpl-violations.org` project has detected license compliance problems on over 150 products, such as a Linksys router [7]. It is possible that the developers working in these situations knowingly used the licensed software inappropriately. However, it is also possible that the developers did not understand the implications of using the open source software as they did.

Despite an indication that license problems occur when using open source software components, there is little research investigating whether developers understand the licenses and how to use them. We could find evidence of only one investigation of license interactions from a developer's point of view. Vendome and colleagues surveyed developers from projects in which licenses for software changed, asking how licenses were picked and reasons for license evolution [8]. The focus of their study was thus on developers involved in license change decisions as opposed to developers involved in using components.

In this paper, we directly explore whether developers involved in open source understand licenses and their interactions. We report on the results of a survey that asked developers about 42 different cases of the use of code under different open source licenses. To make the survey tractable for developers to answer, we focused on three popular open source licenses (GNU GPL 3.0, GNU LGPL 3.0 and MPL 2.0). We advertised the survey on mailing lists and Twitter and collected responses from 375 participants, who were largely developers and who came from many parts of the world.

978-1-5386-0535-6/17 $31.00 © 2017 IEEE
DOI 10.1109/ICPC.2017.7

Analyzing the survey responses required determining, for each case, appropriate answers. We determined these answers by recruiting an intellectual property lawyer with deep knowledge of the open source community as our oracle. This expert, who has over a decade's specialization in patent reform, open source licensing, and related issues, kindly gave us his opinion on each of the scenarios we constructed, and helped us identify ambiguities and missing information so that we could check to see if our developer respondents spotted the same issues and analyzed them the same way.

Our analysis of the survey responses indicate that developers had a good grasp on development cases involving a single license. However, developers struggled when more than one license was involved. Developers recognized that some license interaction cases were more dependent on technical details and others on license details, but they lacked a deep grasp of the intricacies of license interactions.

This paper makes three contributions:

- It provides empirical evidence that software developers understand how to use individual open source licenses in both simple and complex development scenarios.
- It provides empirical evidence that software developers struggle to understand cases involving combinations of open source licenses.
- It provides an analysis of the factors that developers consider as they work with combinations of open source licenses.

The results of the survey indicate a need to help software developers better understand the ramifications of the licenses associated with open source software components that they rely upon. This help needs to include tool support that can help a developer comprehend, and reason about interactions between licenses in the context of how components are being incorporated and modified into the software being built. There is also potentially a role for recommenders that can recommend how code should be structured to enable the use of components with particular license characteristics.

We begin with an overview of previous related research on open source licenses (Section II) and a brief overview of the licenses used in the survey (Section III). We then describe the survey, (Section IV), our analysis methods (Section IV) and present the results (Section V). The latter parts of the paper describe threats to the results (Section VI), discuss ways forward to improve the situation (Section VII) and summarize the paper (Section VIII).

II. Related Work

The role of open source licenses on open source projects has been the focus of research from a number of different perspectives.

Some researchers have focused on the overall trends of open source license use. For example, Aslett has shown the ratio of permissive (e.g., MIT style licenses that have limited restrictions on reuse) vs. restrictive (e.g., GPL style licenses that require changes to be open source) licensed projects shifted in favour of permissive licenses between 2008 and 2011 [9]. This result is echoed in the work of Hofmann and colleagues [10]. Di Penta and colleagues have looked at the issue of open source license use at a more detailed level, considering how licenses can change for and within a project over time [11]. By considering overall trends, these researchers have helped provide a characterization of open source license use, showing that many licenses are used in practice and that the choice of a license is not static for a project as a whole or for parts of a project.

Other researchers have taken a different perspective, considering how the choice of a particular open source license (or licenses) can impact a project. Through analysis of open source project artifacts, Stewart and colleagues found business-friendly open source licenses had a positive association with project success [12], where success is defined as user interest in and development activity on a project. Using a similar analysis approach, Sen and colleagues determine factors that affect the choice of a license for a project (e.g., [13]).

Another area of research focus has been on the impact of license interactions on software development. Germán and Hassan were the first to describe the license mismatch problem, which occurs when two or more pieces of software with different licenses with different restrictions are combined in a new project. Germán and Hassan developed a model to describe mismatch problems and to document integration patterns for solving such problems [14]. Researchers have since looked at the license mismatch problem in several different ways. For example, Alspaugh and colleagues developed a meta-model to analyze the interaction of licenses from the viewpoint of software architecture [15]. Germán and colleagues developed the Ninka tool [16] to identify licenses in source code, making possible larger scale studies of license use and evolution (e.g., [17]). The research in this area has focused largely on technical aspects and implications of licenses, such as the detection of license interactions and the role of software structure both in causing inappropriate interactions and ways to resolve interactions.

In this paper, we focus on the developer perspective on open source licenses, considering how well developers understand open source licenses, particularly when those licenses interact. In considering the developers' perspectives, our work is closest to Vendome and colleagues [8]. Their work includes a survey of 138 developers chosen from projects in which the evolution of a license use occurred. Their survey focused on how developers picked licenses and motivations for license changes. This survey relies on the fact that developers responding to the survey understood the licenses with which they work and the situations in which the licenses are used. In this paper, we focus on a more general population of developers, not just those who have dealt with license changes, and delve into the question of whether the more general population understands open source license use both when one license is used and when a combination of licenses is in use.

III. Overview of Licenses

Before we introduce the method we used to investigate questions regarding developer knowledge of the use of open source licenses, we present a brief overview of the licenses referred to in the survey. We focused on these licenses because they represent common licenses in use (e.g., [8]), because they represent a range from restrictive (e.g., GNU GPL) to permissive (e.g., Mozilla Public License (MPL)) and because they represent different technological choices in license application and resultant restrictions (e.g., GNU GPL vs. GNU LGPL).

a) GNU General Public License (GPL), Version 3.0 (GPL-3.0): The GNU General Public License (GPL)[1] ensures end users of the software being licensed will be able to run, view, share and modify the software. The GPL is a copyleft license that requires the rights to be retained when software is shared or modified. The updates to the GPL in Version 3.0 were instituted to protect the copyleft features given recent legal and technological changes.

b) GNU Lesser General Public License, Version 3.0 (LGPL-3.0): The GNU Lesser General Public License (LGPL)[2] is a weak copyleft license. The LGPL can be applied to software that is deployed as a shared library; code in the shared library must be available to be viewed, modified and shared, but proprietary code using the library need not be made freely available.

c) Mozilla Public License, Version 2.0 (MPL-2.0): The Mozilla Public License (MPL-2.0)[3] provides a different balance between proprietary and free software. The MPL-2.0 is copyleft, similar to the GPL-3.0, but at the file level, easing the combination of code under different licenses. For example, software that is a mixture of both proprietary and MPL code requires only modifications to files licensed under the MPL to be made available.

IV. Method

We chose a survey instrument to investigate whether developers understand open source licenses because we were interested in trends about which aspects of licenses developers understand and which aspects developers struggle with. Trends identified in a survey can later be investigated in more depth using other instruments, such as interviews.

To develop the survey, two authors of this paper posited a number of different cases of how software might be licensed, how a software system might be built out of existing components and how software might evolve. The other two authors of the paper, each of whom has extensive development experience, commented and helped refine the cases. All authors of the paper then collaboratively developed a set of scenarios based on the identified cases. We originally planned to include four licenses in our survey. After piloting, we decided to reduce the number of licenses (to 3) and scenarios (to 7) to keep survey completion time under 40 minutes.

[1]https://opensource.org/licenses/GPL-3.0
[2]https://opensource.org/licenses/LGPL-3.0
[3]https://opensource.org/licenses/MPL-2.0

A. Survey

The on-line survey we distributed consisted of:

- six demographic questions,
- seven hypothetical software development scenarios, some of which included multiple license combinations, and
- four open-ended questions.

A copy of the full survey is available for reference.[4]

Table I summarizes the seven development scenarios in the survey. The first scenario description in Table I is laid out similar to how the questions appeared in the survey. The second scenario description shows how questions were posed about different combinations of licenses. The remaining scenario descriptions are in compact form. These scenarios included a total of 45 cases. In this paper, we refer to the cases by their scenario number (as in S1) and the license (or licenses involved, as in S2-GPL-GPL). For each case, a participant could answer `yes`, `no` or `unsure`. If a participant answered `unsure`, an extra textbox appeared after the question asking for further clarification as an open-ended text comment. For each scenario, there was also an open-ended text box for the participant to state any assumptions she or he made about the scenario posed.

In developing the survey, we had to make a choice between full and long-specifications of all the details of a given scenario, such as detailed descriptions of the architecture of the software being combined in some scenarios, versus more brief descriptions. We chose the latter approach to make the survey tractable for participants and as we were interested in the assumptions participants may (or may not) make.

B. Participant Recruitment

We recruited participants in two ways. First, three authors of the paper, with over 6,000 followers combined, tweeted about the survey with a link to the survey. Second, the authors used mailing lists to advertise the survey.

C. Analysis

We analyzed the results of the survey both quantitatively and qualitatively.

Quantitative analysis required *correct* answers for each of the 45 cases. As described in Section I, we asked a lawyer with relevant expertise to rule on each case and to point out ambiguities and omissions that could affect the answer. This allowed us to score responses from developers and also to check how many were able to correctly identify the same issues.

Qualitative analysis was performed on the open-ended comments provided by participants about answers for cases for which they were unsure and assumptions made for the scenarios in which the case appeared. Two of the authors open-coded the collected comments [18]. We focused on comments and assumptions for questions for which over 30% of the answers did not match the legal expert's answer (14 cases) or for which over 10% of the answers were "unsure" (14 cases). We chose

[4]https://goo.gl/v2JGol

TABLE I
SURVEY SCENARIOS

Scenario #1 - Layout similar to survey

John has been working on `ToDoApp`, his own personal task management application. `ToDoApp` is going to be a desktop-based application that will be used exclusively by John on his own computer. To make sure he does not lose any of his very special tasks, John is planning to use a lightweight library called `LightDB` to persist `ToDoApp`'s data.

If `LightDB` is distributed under the following licenses, would John be allowed to use it as part of `ToDoApp`?

GNU GPL3.0 (*S1-GPL*)	◦ Yes	◦ No	◦ Unsure
GNU LGPL3.0 (*S1-LGPL)*	◦ Yes	◦ No	◦ Unsure
MPL 2.0 (*S1-MPL)*	◦ Yes	◦ No	◦ Unsure

Scenario #2 - Layout abbreviated but similar to survey

Having used `ToDoApp` for three months, John realized how much his productivity has improved. To help other people manage their tasks as efficiently as well, John has decided to make `ToDoApp` available as open source.

If `LightDB`, the lightweight library used to persist `ToDoApp`'s data is distributed under **GNU GPL 3.0** would John be allowed to make `ToDoApp` available under the following licenses?

GNU GPL3.0 (*S2-GPL-GPL*)	◦ Yes	◦ No	◦ Unsure
GNU LGPL3.0 (*S2-GPL-LGPL)*	◦ Yes	◦ No	◦ Unsure
MPL 2.0 (*S2-GPL-MPL)*	◦ Yes	◦ No	◦ Unsure

Survey repeats the question for `LightDB` under **GNU LGPL 3.0** and **MPL 2.0** for each license combination for `ToDoApp`.

Scenario #3 - Compact Form

After the success of the open source version of `ToDoApp`, John has decided to create a brand new commercial task management application: `TaskPro`. `TaskPro` is going to be built from scratch and use `LightDB` as a lightweight library to persist data.

If `LightDB`, is distributed under {**GNU GPL 3.0**, **GNU LGPL 3.0**, **MPL 2.0**}, would John be allowed to make `TaskPro` commercially available under each of the {**GNU GPL 3.0**, **GNU LGPL 3.0**, **MPL 2.0**} licenses?

Scenario #4 - Compact Form

As the lead developer of a new product at GreatSoftware Inc., Laura decided to use an existing authentication library she found on the Web called `SafeAuth`. She realizes that `SafeAuth` could be improved using a stronger cryptographic algorithm when storing users' information. The product is going to be released under a commercial software license, but Laura would like to release the improved version of `SafeAuth` as open source.

If `SafeAuth`, is distributed under {**GNU GPL 3.0**, **GNU LGPL 3.0**, **MPL 2.0**}, would Laura and her team be allowed to release the improved version of `SafeAuth` under each of the {**GNU GPL 3.0**, **GNU LGPL 3.0**, **MPL 2.0**} licenses?

Scenario #5 - Compact Form

Laura who works for GreatSoftware Inc. has changed the open version of `SafeAuth` found on the Web and added a new, stronger cryptographic algorithm to it. Despite Laura's intentions to release the modified version of `SafeAuth` as open source, her manager sees a very strong competitive advantage for their products and decides not to release the modified version as open source.

Considering that the new product is going to be distributed under a commercial license, if `SafeAuth` is distributed under the {**GNU GPL 3.0**, **GNU LGPL 3.0** and **MPL 2.0**}, can Laura and her team use the modified version as part of their new product?

Scenario #6 - Compact Form

Shaoqing believes there are unhappy users out there willing to pay for a premium email client. To get to market faster, she decided to use an open source implementation of the `Simple Mail Transfer Protocol (SMTP)`.

If the `SMTP` implementation is released under {**GNU GPL 3.0**, **GNU LGPL 3.0**, **MPL 2.0**}, would Shaoqing be allowed to fork the `SMTP` project and change the fork's license to the {**GNU GPL 3.0**, **GNU LGPL 3.0**, **MPL 2.0**} license in order to use it in her commercial e-mail client??

Scenario #7 - Compact Form

Shaoqing has been trying to optimize the way her email client handles old e-mails. Browsing on the Web, she found a fairly sophisticated implementation of a compression algorithm on a software developer's blog that could be used on archived emails. The algorithm implementation has hundreds of lines of code and does not include an explicit license, but there is a copyright notice on the blog that states "All Rights Reserved".
If Shaoqing used the source code she found on the blog in her e-mail client, would be allowed to distribute the e-mail client commercially under the {**GNU GPL 3.0**, **GNU LGPL 3.0**, **MPL 2.0**} license?

30% as the threshold for which answers did not match the legal expert's answer to allow room for ambiguity; we believe a simple majority threshold would have suggested the answers are always clearcut. We choose to analyze cases for which over 10% of the answers were "unsure" to capture where ambiguity seemed to be likely occurring. Applying these thresholds, nine out of the 28 cases identified overlapped, resulting in the coding of comments for 19 cases and assumptions for

four scenarios. We took the approach of assigning only one code to each comment, choosing the code that best described the comment. For comments for five of the cases (a total of 132 comments), each coder independently coded the case and then the coders met to discuss the codes and form agreement. After five cases were coded, the coders independently coded the remaining 14 cases and then met to form an agreement. The Cohen's kappa score for the 14 cases that involved 393 comments was .836. Similarly, the coders independently coded assumptions for one scenario (a total of 53 comments), met to form agreement and then independently coded the assumptions for the remaining three scenarios. The Cohen's kappa score for coding assumptions for the 282 assumptions across the three scenarios was .853.

V. Results

The survey was started 825 times. Ultimately, 375 individuals completed the survey for a completion rate of 45%. We report on the demographics of the participants before providing a quantitative and qualitative analysis of the respondents' results. A data package containing aggregate results and participants' comments may be consulted for further information.[5]

A. Participants

Participants came from all over the world (i.e., USA (267), UK (133), Germany (63), Canada (62), etc.). Table II summarizes the roles and experience of the participants. As the majority of the participants identified their role as developer or team lead, we will refer to our findings from the survey in terms of findings about developers. The participants had significant software development experience, with 93% reporting at three years or more. The participants came from a range of size of company with 20% working in companies with over 5000 employees down to 7% working solely. Most participants have previously chosen software licenses and most contribute to open source. The table shows the top five languages used, with participants reporting use of over 10 different programming languages.

TABLE II
Participant Demographics

Demographics	
Job title	
Programmer/Soft. Dev./Soft. Eng.	67.2%
Technical Lead/Team Leader	13.3%
Other	13%
Sys. Administrator/Network Engineer	3.2%
Project Manager	2.9%
Level of Experience	
More than 7 years	61.3%
At least 3 years	32.3%
Less than 3 years	6.4%
Ever chose a software project's license?	
Yes	85.3%
No	14.7%
Often contribute to open source projects?	
Yes	74.7%
No	25.3%
Programming languages used the most	
Python	51.6%
JavaScript	25.7%
C++	25.4%
Java	22.5%
C	19.8%

B. Quantitative Results

Table III presents the participants' survey answers. For each case, a bar chart is shown that compares the participants' responses for the case to the legal expert's opinion: a dot is used to indicate the response of the legal expert in each case. For each chart, the green bar represents "Yes" answers, the red bar represents "No" answers and the blue bar represents "Unsure" answers.

As we delved into the details of participants' responses, we noted an issue with Scenario 5. By comparing the assumptions of the legal expert with the participants' assumptions, we determined that the scenario was not stated clearly enough in terms of which source code would eventually be released. As a result, we have omitted the results for Scenario 5 in Table III and we do not include Scenario 5 in any further reporting or analysis. Its removal leaves 42 cases for analysis.

We consider that the participants' answers for a case are correct when 70% or over of the participants' answers match the legal expert's. We chose this threshold to account for the potential ambiguity in our short scenario descriptions, which may lead participants to be unsure of the meaning of the scenarios. Applying this threshold, participants' responses matched the legal expert's in 26 of the 42 cases (62%). This rate of matching the legal expert's opinion is encouraging as it suggests that participants understood many aspects of the open source licenses used in the scenarios. Participants also matched the opinion of the legal expert whenever only one open source license is in use in the scenario (e.g., S2-GPL-GPL or S7-MPL-MPL, etc.). However, when more than one license is involved, the participants' answers differed from the expert's. We were not able to find any trend in particular license combinations that were troublesome: four cases involved GPL and LGPL, three cases involved GPL and MPL and five cases involved LGPL and MPL. From these results, we make the following two observations.

> *Observation 1*. Developers cope well with single licenses even in complex scenarios.

> *Observation 2*. Developers have difficulty interpreting which actions are allowed in scenarios where more than one open source license is in use.

To learn what aspects the participants struggled with, we focus our remaining analysis on the 12 cases in which the participants answered differently than the legal expert and the

[5]https://www.cs.ubc.ca/labs/spl/projects/softwarelicensing/resources/UBC_SPL_software_licensing_survey_data.zip

TABLE IV
CODES FOR ASSUMPTIONS

AG	Authorship/Geo	Who is author and where are they located
CD	Change Dependent	What system files are to be modified
I	Invalid	Text could not be interpreted
IQ	Invalid Question	Participant felt question was invalid
LA	License Assumption	Characteristics of licenses
LI	License Interactions	Ramifications of more than one license
TA	Tech. Assumption	System structure, deployment, etc.
PA	Patent Assumption	Patent or IP
SC	Specific Case	Dual licensing
TeA	Term Assumption	Meaning of term unclear in license or scenario
U	Unsure	Unsure about scenario or license

13 cases for which over 10% of the participants were not sure which answer to choose. These cases overlap, resulting in 17 cases on which we focus our remaining analysis.

C. Qualitative Analysis

The qualitative analysis we conducted considered assumptions at the scenario level and comments made by participants about individual cases.

1) Assumptions: For each scenario, participants had the opportunity to express assumptions that they made when answering the cases. We coded the assumptions for the three scenarios (S2, S3 and S4) on which we focus our qualitative analysis.

We ended up defining 11 codes to describe the assumptions; Table IV describes these codes. The codes demonstrate the wide range of concerns participants considered when thinking about the use of the open source licenses. For instance, participants thought deeply about how the nature of the change—which files (CD), who is making changes (AG), where the author was located (AG), effects on the product architecture (TA)—could affect the situation. Participants also thought deeply about how licenses might interact, such as how code under one license might be re-licensed or dual licensed (LI), and whether the licenses in use included particular exhibits, such as Exhibit B for the MPL that enables an author to mark certain files as incompatible with secondary licenses (LA). Other codes capture comments that we could not interpret (I), such as *"All's good"*, descriptions of why the participant felt the question was invalid (IQ), where participants questioned the meaning of terms in the question (i.e., TeA) and where participants were unsure, such as not knowing a license being asked about (i.e., U).

Table VI states the frequency of each assumption code and indicates the total number of assumptions received for each scenario (ranging from 14% of the participants stating assumptions for S4 to 30% of the participants stating assumptions for S3). For scenario 2, assumptions about the technical aspects of the scenario were most frequent, with over 42% of the assumptions having the TA code. This scenario involved an open source application that might be licensed differently than an open source library that the application relies on. Participants considered such questions as how the library might be used; for instance, would the library be statically or dynamically linked to the application?

Participants also questioned how the code would be structured and released for scenario 3 (52% of codes were tagged with the TA code), in which participants were asked to consider a commercial distribution of a product including open source. For this scenario, participants differed in response from the expert in three of nine cases, with two of those cases involving MPL which describes license constraints at the file level. The difference in participant responses from the expert may also be related to the larger number of questions by participants about terms used in the scenario, such as the precise meaning of the term "commercial"; the term assumption code (TeA) accounted for 22% of the codes for this scenario.

For scenario 4, which described a developer making modifications to an open source library and releasing the modified version of the library as open source, the most frequently occurring code is about the nature of the change, specifically which and how many files might be changed (over 18% of the codes are CD). Participant responses for three of the nine cases in this scenario did not match the legal expert's opinion; all cases that differed involved the GPL and LGPL licenses, including how they interact with each other and how they interact with MPL. Although many participants understood that MPL places file-based restrictions on subsequent use when modified, the majority of participants thought there were ways to structure the modifications to enable MPL licensed code to be redistributed as GPL or LGPL code (S4-MPL-GPL and S4-MPL-LGPL).

For scenarios 2 and 4, the second most frequently occurring group of codes relates to licenses. For scenario 2, 28% of the assumptions questioned aspects of the licenses, such as what relicensing is possible for source under a given open source license and the possibilities for dual licensing code (i.e., the LI and LA codes). Over 20% of the assumptions for scenario 4 relate to licenses (i.e., 20.8% of codes are LA and 7.3% of codes are LI). Some assumptions with these codes indicate significant knowledge of one or more of the open source licenses used in the scenarios, such as referencing the "tri-license" header.

This analysis leads us to two additional observations.

> *Observation 3.* Developers understand technical decisions will impact open source license use.

> *Observation 4.* Developers recognize that there are interactions between open source licenses, but those interactions were not always correctly interpreted.

2) Case Comments: We also analyzed 462 comments provided by participants for the 17 cases of interest. Table V describes the seven codes that resulted from the process described in Section IV-C; these codes include comments

TABLE III

PARTICIPANT RESPONSES BY CASE. DOTS INDICATE ANSWER OF LEGAL EXPERT. GREEN BAR FOR EACH CASE IS A "YES" ANSWER; RED BAR IS A "NO" ANSWER AND BLUE BAR IS A "UNSURE" ANSWER.

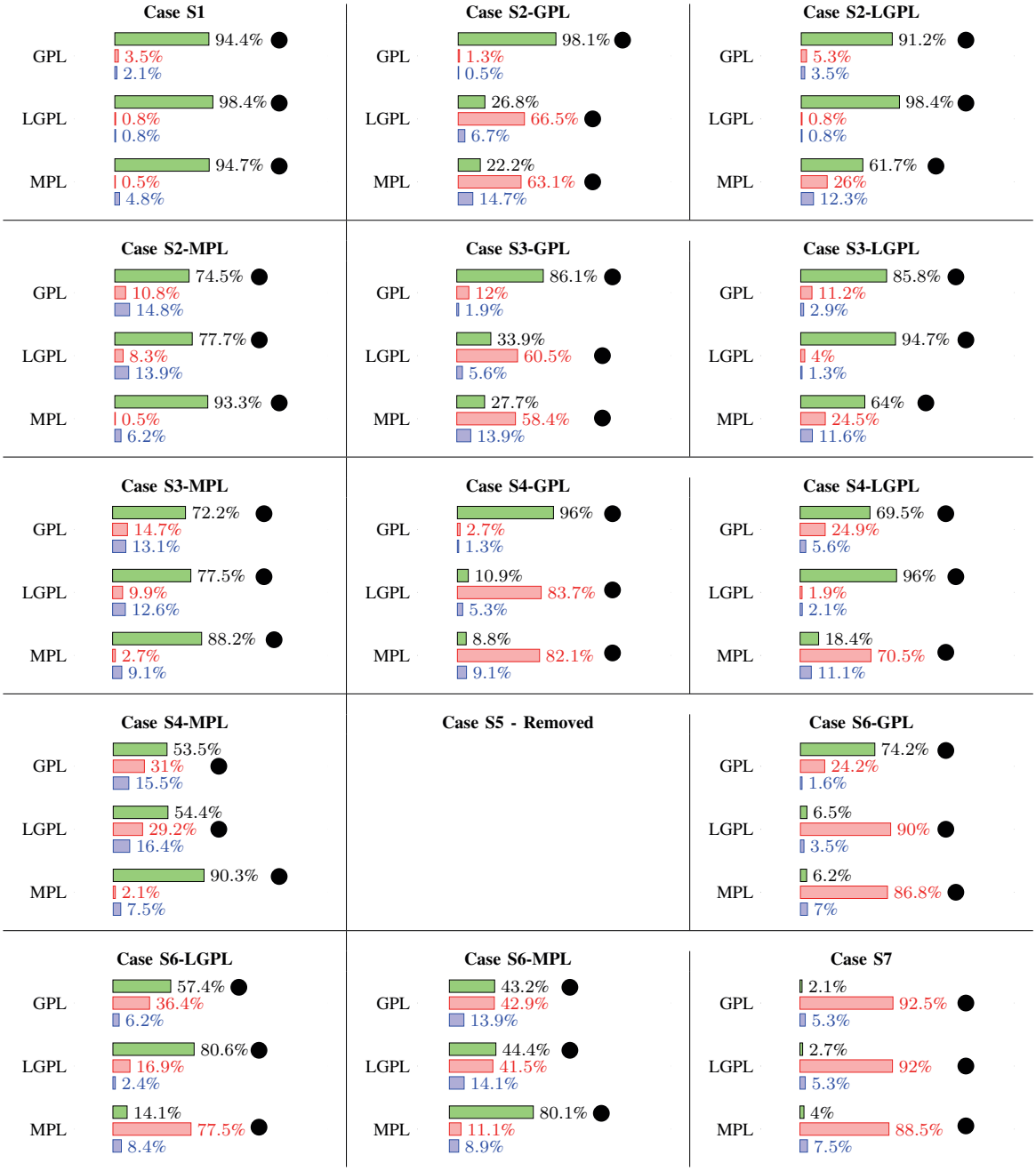

TABLE V
CODES FOR COMMENTS FOR CASES

A	Assumption	An assumption about the case.
Am	Ambiguity	Description of ambiguous point in case.
I	Invalid	Meaning of comment unclear.
LI	License Interaction	Concern about actions possible with more than one license.
SC	Specific Case	Concern about relicensing or dual license.
TD	Technical Detail	Concern about technical aspects of case.
U	Unsure	Case is unclear.

about license interactions (LI), specific cases of relicensing or dual licensing (SC), technical details regarding the scenario (TD), and uncertainty about the scenario (U, A, Am and U).

Table VII reports on the frequency of occurrences of each code in the comments for each case. From this table, it becomes evident that different combinations of codes appear for the same license combinations: the S2-GPL-LGPL case, as an example, has 20% of comments as license interactions, but license interactions were not a concern for the S3-GPL-LGPL and S6-LGPL-GPL cases. These three cases differ in how the software is used, changed and combined (i.e., the technical context), leading to the following observation.

> *Observation 5.* Questions that arise about the use of multiple open source licenses are situationally dependent.

The most frequently occurring code across the cases is the Unsure code. This uncertainty often had to do with details related to the licenses, such as "don't know if GPL allows it" and "don't know to which point GPL is viral". The prevalence of this code leads to the following observation.

> *Observation 6.* A number of developers lack knowledge of the details of open source licenses.

The second more frequently occurring code across the cases studied was for license interactions (LI). Comments made by participants echoed concerns raised in the assumption coding, such as which licenses could subsume other licenses, when code could be dual licensed and when code could be relicensed. This data helps reinforce *Observation 4*.

D. Detailed Examples

We look at two cases in more detail to provide more context for the findings that participants struggled with license interactions and technical assumptions. The two cases we consider are from scenario 2, which involves the use of an existing open source library (LightDB) to build an application that will also be released under an open source license (ToDoApp). A full description of scenario 2 is available in Table I.

1) License Interaction: For the S2-GPL-MPL case, only 63.1% of participants responded with an answer that matched the legal expert's opinion and 14.7% of the participants were unsure.

After the unsure code, the second most frequently occurring code for this case was the license interaction (LI) code. In these comments, participants expressed their doubts about appropriate interactions. For instance, one participant wrote: *"I don't understand how the secondary license restriction and GPL interact"*. Another wrote: *"MPL/(L)GPL dual licensing is popular, so I assume there is a reason for that"*. Even in a comment labelled unsure, a participant recognized license interactions might be relevant: *"Have not studied the details; generically expect trouble when mixing non-GPL licenses with GPL so would have guessed 'No' if forced"*. These comments highlight that although participants recognize license interaction, they do not understand the intricate details of when interactions occur or the results.

2) Technical Details: For the S2-GPL-LGPL case, only 66.5% of participants responded with an answer that matched the legal expert's opinion and 12.3% were unsure.

After the unsure code, the most frequently occurring code was the technical details (TD) code. Participants expressed a need to understand more technical details about the scenario in order to interpret how the licenses would interact. For example, one participant wrote: *"It depends on how ToDoApp is distributed. If ToDoApp was only distributed as source then this would be fine. For binary distributions, if ToDoApp is statically linked against LightDB it must be distributed under GPL. The case is less clear for dynamically linked code - I understand the FSF and other organizations disagree!"*. This comment indicates knowledge of the licenses and views of the communities around the licenses. Other participants knew the technical details might matter, but not why: *"I think it might depend on how the two libraries are linked together"*.

TABLE VII
CODES DESCRIBING CASES

Case	Code (%)							
	A	Am	I	LI	SC	TD	U	#
Scenario 2								
S2-GPL-LGPL	-	-	-	20.0	-	32.0	48.0	25
S2-GPL-MPL	-	-	2.2	17.4	6.5	8.7	65.2	46
S2-LGPL-MPL	-	-	5.9	14.7	2.9	11.8	64.7	34
S2-MPL-GPL	-	2.5	7.5	25.0	7.5	-	57.5	40
S2-MPL-LGPL	-	5.6	-	19.4	5.6	-	69.4	36
Scenario 3								
S3-GPL-LGPL	-	-	46.7	-	-	26.7	26.7	15
S3-GPL-MPL	2.9	-	11.4	8.6	5.7	8.6	62.9	35
S3-LGPL-MPL	-	-	8.0	12.0	4.0	4.0	72.0	25
S3-MPL-GPL	3.5	-	6.9	17.2	3.5	3.5	65.5	29
S3-MPL-LGPL	-	3.6	3.6	17.9	3.6	3.6	67.9	28
Scenario 4								
S4-LGPL-GPL	-	-	15.4	15.4	46.2	-	23.1	13
S4-LGPL-MPL	-	-	5.6	33.3	5.6	5.6	50.0	18
S4-MPL-GPL	-	3.3	10.0	23.3	13.3	-	50.0	30
S4-MPL-LGPL	-	6.1	15.2	21.2	12.1	-	45.5	33
Scenario 6								
S6-LGPL-GPL	-	7.1	7.1	-	-	-	85.7	14
S6-MPL-GPL	-	-	-	9.5	33.3	-	57.1	21
S6-MPL-LGPL	-	-	-	10.0	10.0	-	80.0	20

TABLE VI
CODES DESCRIBING ASSUMPTIONS

Scenario #	Code (%)											Total
	AG	CD	I	IQ	LA	LI	PA	SC	TA	TeA	U	#
Scenario 2	2.1	5.2	6.3	3.1	20.8	7.3	2.1	4.2	42.7	1.0	5.2	96
Scenario 3	0.9	1.8	4.4	0.9	8.9	2.7	0.9	-	52.2	22.1	5.3	113
Scenario 4	3.8	18.9	15.1	7.6	13.2	9.4	5.7	3.8	11.3	5.7	5.7	53

E. Summary of Results

The software developers who took our survey were able to correctly interpret a variety of simple and complex development scenarios involving one license (*Observation 1*). These software developers understand that how the software is built affects license interactions, but they have neither a consistent and deep grasp of what technical details matter (*Observation 2* and *Observation 3*) nor a solid understanding of the intricacies of how licenses interact (*Observation 4*). Developers are aware that different characteristics matter in different situations of multiple license use (*Observation 5*), but overall lack the knowledge to tease apart license interactions across multiple situations (*Observation 6*).

VI. THREATS TO VALIDITY

The survey provided links to the licenses referred to in the survey but did not require participants to answer questions to validate their understanding of individual licenses, which may have affected the construct validity of the survey. We made this choice for two reasons. First, we wanted to allow participants to interact with the licenses as they normally would; for instance, some participants might rely on their knowledge of the licenses, some might reference the licenses to answer survey questions and others might use other sources, such as `choosealicense.com` or knowledgeable colleagues. The survey asked if participants used additional resources: 36% reported using resources such as Wikipedia, `TLDRlegal.com` and `choosealicense.com`.

Second, the overall survey is lengthy and adding more questions to validate understanding of each of three licenses would be even more time consuming for participants. The choice not to validate individual license understanding may have resulted in participants answering questions for which they have no background. For some survey questions, we received a large number of comments, such as over 100; the insight in many of these comments suggests many participants had sufficient background to answer the questions posed. A large number of individuals, 825, started the survey with 45% completing the survey; some of the individuals that started the survey but did not finish might represent individuals without sufficient license knowledge.

The construct validity may also have been affected by the particular three licenses we chose to use in the survey. We deliberately picked a mixture of restrictive (i.e., GPL) and permissive (i.e., MPL) licenses to trigger license interactions. Our findings might differ if only a set of more permissive licenses were used. Future work should investigate developers' understanding of fine-grained license interactions.

The survey has limitations with regards to content validity, which considers the degree to which the survey investigates developers knowledge of open source license use. The survey questions required participants to understand individual licenses, such as GPL, and how the use of the license affects scenarios involving interactions with other licenses. Furthermore, because the questions were presented as multiple choice problems, they are likely not as complex as many of the scenarios faced in practice. As noted above, the survey is limited in what can be concluded about the knowledge of individual open source licenses.

Another issue we faced in the design of the survey was the specificity to provide in the scenarios posed in the survey questions. As some of the participants noted, the wording of the scenarios had some ambiguity. In particular, participants struggled with the wording in scenario 3 in which the term "commercial" was used; the confusion involved whether the term implied any changes made to open source software were to be kept as closed source or whether the term meant money may be charged for use of the resultant software. We did not foresee this ambiguity and note that the term commercial has also been used in previous surveys on open source software [8]. The ambiguity also did not arise in pilots we conducted of the survey questions. The lack of specificity may have also caused differences between the legal expert's reading of the cases and the participants'. We have tried to mitigate the effects of question ambiguity through careful analysis of the legal expert's input and careful analysis and reporting of the qualitative comments provided by participants.

As described in Section IV, the participants in the survey came from a large number of countries, used a wide variety of programming languages and largely described their job as a software developer. As noted above, the techniques we used to recruit participants for the survey may have biased the population from which the participants are drawn. In particular, we observed that the large majority of the participants (85.3%) had chosen a software project's license before, which might be an instance of self-selection bias. The diversity of participants suggests the results may be applicable to a reasonable segment of open source developers.

VII. DISCUSSION

The survey results indicate that most of the 375 respondents to our survey struggle with understanding the interaction of open source licenses in both simple and complex software development cases. These situations can arise often. As stated in Section I, Sonatype reports that most Java applications built using Maven incorporate over 100 open source components;

it is highly unlikely these components will all have the same open source license. Furthermore, as indicated by comments of our participants, in some cases, whether or not licenses are compatible depends on the structure of the code using the components.

Tool support is needed to help developers deal with license incompatibilities when using open source components. Germán and Hassan [14] provides a model for identifying possible mismatches when different open source licenses interact, but is not able to recognize code structures that cause these mismatches. Vendome goes further, suggesting a need to find incompatibilities, explain why there is an incompatibility and recommend a way to fix incompatibilities, possibly through a license change or through code restructuring [19]. The comments we analyzed from participants completing our survey suggest the recommendation engine may need to be more extensive and robust than suggested by Vendome. Participants described ways to restructure their own code and/or change the open source code to enable the use of a component without causing license incompatibilities. This type of recommender may require a means of formally modelling licenses, the effects of a license in terms of how it is used in code, and the variability allowed by the license in terms of code interactions. This formal model would then need to be potentially integrated with code refactoring tools, to perhaps automatically search for refactorings that would allow a license compatibility check to pass. Models, such as that introduced by Alspaugh and colleagues [15], may provide a starting point for building such tools.

VIII. Summary

Open source software is not a self-contained world with a specific set of developers involved and a small set of open source licenses with well-defined interactions. Many closed-source, commercially-oriented software projects rely on open source software. Many open source licenses exist with different ramifications depending on how the software with different licenses interact (i.e., via dynamic linking, copying of source code, etc.). Many software developers work on a variety of open source projects with different licenses and move back and forth between open and closed source software projects.

The results of our survey indicate that many of the 375 respondents to our survey, who were largely software developers, have a good grasp of at least three open source licenses when only one of those licenses is being used. When a combination of open source licenses is being used, developers struggle to ask the right questions for the situation, such as whether to focus on technical details of the situation or generic issues of how two licenses interact (i.e., is one license more permissive than another). Overall, our survey indicates that the developers who responded lack the knowledge and understanding to tease apart license interactions across multiple situations.

This survey is the first to our knowledge that has delved into developers' understanding of open source licenses as the one previous survey has targeted developers specifically involved in license changes to open source software. Our exploratory survey aims to investigate trends in comprehension of open source licenses, rather than suggest specific changes to the licenses studied. Our results do indicate the need for tool support to help developers overcome license incompatibilities.

Given the importance of open source software to many of the software products being built, we hope these results motivate others to also learn more about how software developers interact with open source licenses and to build tool support to help developers comprehend and work with this information that is a critical part of the growing number of software components available for use.

Acknowledgment

We would like to thank the participants of the survey for the time they spent and their insightful comments. We would especially like to think the legal expert who surveyed as our oracle for survey answers. This work was supported in part by NSERC. We also thank Michalis Famelis for helpful comments on an earlier draft.

References

[1] Sonatype, "2015 state of the software supply chain report: Hidden speed bumps on the road to "continuous"," 2015.

[2] M. Rosson and J. Carroll, "The reuse of uses in Smalltalk programming," *ACM Transactions on Computer-Human Interaction*, vol. 3, no. 3, pp. 219–253, 1996.

[3] S. Clarke, "Measuring API usability," *Dr. Dobb's Journal, Special Windows/NET Supplement*, 2004.

[4] M. Robillard and R. DeLine, "A field study of API learning obstacles," *Empirical Software Engineering*, vol. 16, p. 703, 2011.

[5] C. Vendome, "A large scale study of license usage on github," in *37th International Conference on Software Engineering*, 2015.

[6] D. M. Germán, M. D. Penta, and J. Davies, "Understanding and auditing the licensing of open source software distributions," in *The 18th IEEE Int'l Conf. on Program Comprehension*, 2010, pp. 84–93.

[7] A. Hemel, K. T. Kalleberg, R. Vermaas, and E. Dolstra, "Finding software license violations through binary code clone detection," in *Proceedings of the 8th Working Conference on Mining Software Repositories*. ACM, 2011, pp. 63–72.

[8] C. Vendome, M. L. Vásquez, G. Bavota, M. D. Penta, D. M. Germán, and D. Poshyvanyk, "When and why developers adopt and change software licenses," in *2015 IEEE International Conference on Software Maintenance and Evolution*, 2015, pp. 31–40.

[9] "On the continuing decline of the gpl," http://blogs.the451group.com/opensource/2011/12/15/on-the-continuing-decline-of-the-gpl/, 2011.

[10] G. Hofmann, D. Riehle, C. Kolassa, and W. Mauerer, "A dual model of open source license growth," in *IFIP International Conference on Open Source Systems*. Springer, 2013, pp. 245–256.

[11] M. Di Penta, D. M. German, Y.-G. 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*. ACM, 2010, pp. 145–154.

[12] K. J. Stewart, A. P. Ammeter, and L. M. Maruping, "Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects," *Info. Sys. Research*, vol. 17, no. 2, pp. 126–144, Jun. 2006.

[13] R. Sen, C. Subramaniam, and M. Nelson, "Determinants of the choice of open source software license," *J. Manage. Inf. Syst.*, vol. 25, no. 3, pp. 207–240, Dec. 2008.

[14] D. M. Germán and A. E. Hassan, "License integration patterns: Addressing license mismatches in component-based development," in *Proc. of 31st Int'l Conf. on Soft. Eng*, 2009, pp. 188–198.

[15] T. A. Alspaugh, W. Scacchi, and H. U. Asuncion, "Software licenses in context: The challenge of heterogeneously-licensed systems," *Journal of the Association for Information Systems*, vol. 11, no. 11, p. 730, 2010.

[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*, 2010, pp. 437–446.

[17] C. Vendome, M. Linares-Vásquez, G. Bavota, M. Di Penta, D. German, 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*, ser. ICPC '15. Piscataway, NJ, USA: IEEE Press, 2015, pp. 218–228.

[18] J. Corbin and A. Strauss, "Grounded theory research: Procedures, canons and evaluation criteria," *Qualitative Sociology*, vol. 13, pp. 3–21, 1990.

[19] C. Vendome and D. Poshyvanyk, "Assisting developers with license compliance," in *Proc. of the International Conference on Software Engineering*, 2016, pp. 811–814.