Investigating Software Developers’ Understanding of Open Source Software Licensing

by

Daniel A. Almeida

Bachelor of Computer Science, Universidade Salvador, 2014

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES
(Computer Science)

The University of British Columbia
(Vancouver)

August 2017

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 help understand whether or not developers understand the open source licenses they use, I 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. To understand the context in which licensing issues arise in practice, I analyzed real-world questions posed by developers on online question-and-answer communities. The analysis of these questions indicate that licensing issues can constrain software evolution and technical decisions can have an impact on future licensing issues. Finally, I interviewed software developers in industry to understand how developers reason about and handle license incompatibility in practice. The developers I interviewed are cautious of restrictive licenses. To identify potential licensing issues, these developers rely on licensing guidelines provided by their organization and sometimes use specialized tools to automatically detect licensing issues in their projects. When faced with a situation in which a component that suits their needs is not compatible, developers tend to look for alternative components made available by open source communities. They sometimes leverage the technical architecture of their projects to enable the use of components under restrictive licenses and might rewrite the required functionality if necessary. An analysis of the results indicate a need for tool support to help guide developers in understanding the structure of the code and the technical details of a project while taking into account the exact requirements imposed by the licenses involved.

Lay Summary

Software developers rely on software components created by other developers to build new software applications. These components are made available under specific licenses that specify under which terms and conditions others can use them. Because modern software applications frequently use many components, it is important for developers to understand how they are allowed to use components. This requires a correct understanding of their licenses. In this thesis, I report on the results of a survey that investigates whether software developers understand a number of hypothetical scenarios involving the use of software components under different licenses. The results indicate that developers struggle to interpret what actions are allowed in those scenarios. To understand the contexts in which these issues arise in practice and how developers handle them, I report on the results of the analysis of real-world cases involving software licensing issues and interviews with software developers in industry.

Preface

All of the work presented in this thesis was conducted in the Software Practices Lab at the University of British Columbia, Point Grey campus. All projects and associated methods were approved by the University of British Columbia's Research Ethics Board [certificate #H16-01527].

A version of Chapter 4 has been published [2]. I was involved in concept formation and responsible for all data collection and analysis, as well as manuscript composition. Greg Wilson and Mike Hoye were involved in the early stages of concept formation. Gail C. Murphy was the supervisory author on this project and was involved throughout the project in concept formation, data analysis, and manuscript composition.

Table of Contents

List of Tables

Acknowledgments

I would like to thank my parents for their unconditional support and love. They are examples of integrity, without which no work should ever be done. I also thank my sister for her kindness and support at all times.

I must thank my supervisor, Gail Murphy, for her invaluable guidance and support. She is an example for those who strive for excellence and deeply care about their work. It was my privilege to learn from her.

I thank my friends, both in Brazil and in Canada, for being there for me whenever I needed them. I also thank the colleagues and friends at the Software Practices Lab and at the Department of Computer Science at UBC.

Finally, I must thank my better half, Larissa, for joining me in this adventure. She not only listened to me as I rambled on or ranted about a variety of topics that were not particularly interesting, but supported me through difficult times and celebrated my victories.

DANIEL A. ALMEIDA

The University of British Columbia
August 2017

Dedication

For of Him and through Him and to Him are all things, to whom be glory forever. Amen. (Rom. 11:36)

Chapter 1

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 [15].

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., [13], [4], [12]). 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 [17]. 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 [8]. Hermel and colleagues describe how the gpl-violations.org project has detected license compliance problems on over 150 products, such as a Linksys router [10]. 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. I 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 [20]. The focus of their study was thus on developers involved in license change decisions as opposed to developers involved in using components.

In this thesis, I first explore whether developers involved in open source software development understand licenses and their interactions through a survey. The survey 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, I focused on three popular open source licenses (GNU GPL 3.0, GNU LGPL 3.0 and MPL 2.0). I 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.

Analyzing the survey responses required determining, for each case, appropriate answers. The answers were determined 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 in the survey, and helped identify ambiguities and missing information that could be checked to see if developer respondents spotted the same issues and analyzed them the same way.

An 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.

The survey results indicate that software developers struggle with license incompatibility issues. However, these results do not shed light on the situations in which these issues happen and how software developers reason and handle license incompatibility issues in practice.

To better undertand the context in which licensing incompatibility issues arise, I analyzed real-world questions involving licensing issues. These questions were obtained from a popular network of question-and-answer communities. To gain insight into how software developers reason about and handle license incompatibility situations in practice, I interviewed software developers who have built or are currently building on open source software in their projects.

The analysis of real-world cases shows many situations in which licensing incompatibility can constrain software evolution. In other situations, developers seem to struggle due to previous technical decisions that can have an impact on licensing issues.

The interviews with software developers indicate that they tend to be cautious of restrictive licenses. They rely on licensing guidelines provided by their organization or use specialized tools to identify potential licensing issues in their projects. In a situation in which a component that suits their needs is under an incompatible license, developers tend to look for alternative components provided by open source communities. In some of these cases, they are able to leverage the technical architecture of the project to enable the use of a component or rewrite the required functionality themselves.

This thesis 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, but 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.
- It identifies contextual situations in which license incompatibility issues arise in practice and investigates how software developers reason about and handle these issues.

The results of the survey and analysis of real-world cases and interviews with software developers indicate a need to help developers better understand the ramifications of the licenses associated with open source software components upon which they rely. 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.

I begin with an overview of previous related research on open source licenses (Chapter 2) and a brief overview of the licenses used in this work (Chapter 3). I then describe the survey, including our method and results, and the analysis of real-world cases and interviews with software developers (Chapter 5). Finally, I present my conclusions and discuss ways forward to improve the situation (Chapter 6).

Chapter 2

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 [1]. This result is echoed in the work of Hofmann and colleagues [11]. 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 [6]. 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 [16], 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., [14]).

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 [7]. 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 [3]. Germán and colleagues developed the Ninka tool [9] to identify licenses in source code, making possible larger scale studies of license use and evolution (e.g., [19] and [21]). 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 thesis, I focus on the developer perspective on open source licenses, considering how well developers understand open source licenses, particularly when those licenses interact, and how developers reason about and deal with licensing issues. In considering the developers' perspectives, our work is closest to Vendome and colleagues [20]. 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 thesis, I 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. Furthermore, I analyze real-world cases in which licensing issues arise to understand their situational context and conduct interviews with software developers in industry to understand how developers reason about and deal with license incompatibility issues in practice.

Chapter 3

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., [20]), 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).

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.

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

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

the library need not be made freely available.

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.

[3] https://opensource.org/licenses/MPL-2.0

Chapter 4

Survey

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.

4.1 Method

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.

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.[1]

Table 4.1 summarizes the seven development scenarios in the survey. The first scenario description in Table 4.1 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.

4.1.1 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.

4.1.2 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 Chapter 1, 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

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

Table 4.1: 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?

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 [5]. 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 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.

4.2 Survey 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.[2]

4.2.1 Participants

Participants came from all over the world (i.e., USA (267), UK (133), Germany (63), Canada (62), etc.). Table 4.2 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.

4.2.2 Quantitative Results

Table 4.3 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 4.3 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

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

Table 4.2: 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%

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 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.

4.2.3 Qualitative Analysis

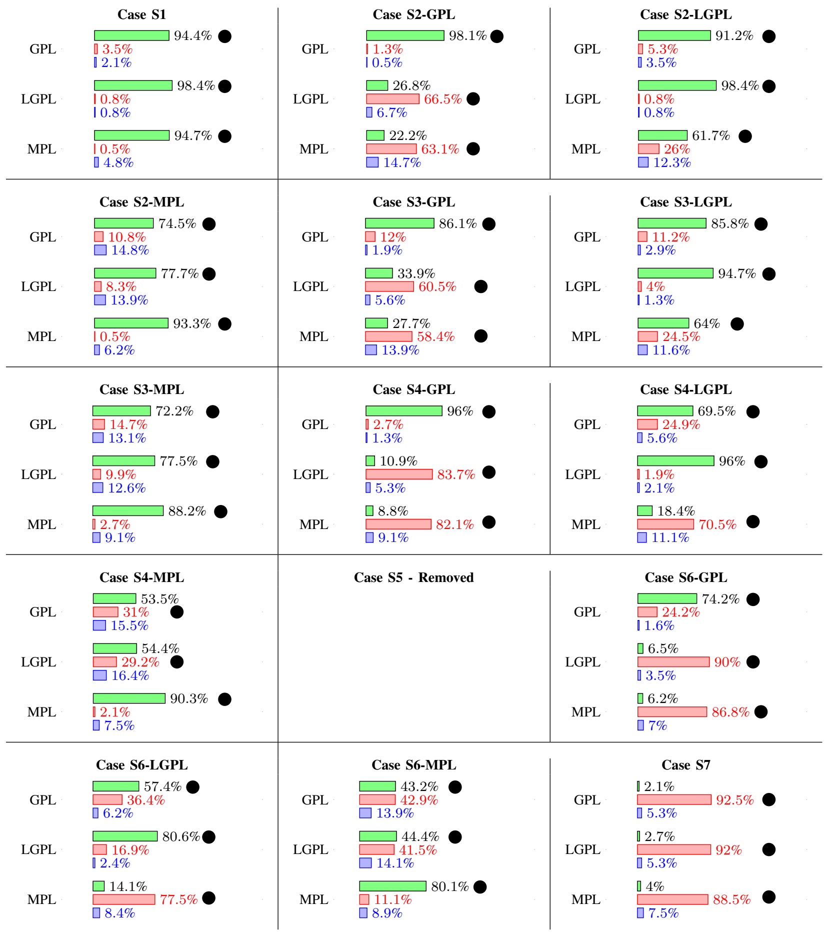
The qualitative analysis we conducted considered assumptions at the scenario level and comments made by participants about individual cases.

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 4.4 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"*, descrip-

Table 4.3: 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.



Case		Yes	No	Unsure
Case S1	GPL	94.4% ●	3.5%	2.1%
	LGPL	98.4% ●	0.8%	0.8%
	MPL	94.7% ●	0.5%	4.8%
Case S2-GPL	GPL	98.1% ●	1.3%	0.5%
	LGPL	26.8%	66.5% ●	6.7%
	MPL	22.2%	63.1% ●	14.7%
Case S2-LGPL	GPL	91.2% ●	5.3%	3.5%
	LGPL	98.4% ●	0.8%	0.8%
	MPL	61.7% ●	26%	12.3%
Case S2-MPL	GPL	74.5% ●	10.8%	14.8%
	LGPL	77.7% ●	8.3%	13.9%
	MPL	93.3% ●	0.5%	6.2%
Case S3-GPL	GPL	86.1% ●	12%	1.9%
	LGPL	33.9%	60.5% ●	5.6%
	MPL	27.7%	58.4% ●	13.9%
Case S3-LGPL	GPL	85.8% ●	11.2%	2.9%
	LGPL	94.7% ●	4%	1.3%
	MPL	64% ●	24.5%	11.6%
Case S3-MPL	GPL	72.2% ●	14.7%	13.1%
	LGPL	77.5% ●	9.9%	12.6%
	MPL	88.2% ●	2.7%	9.1%
Case S4-GPL	GPL	96% ●	2.7%	1.3%
	LGPL	10.9%	83.7% ●	5.3%
	MPL	8.8%	82.1% ●	9.1%
Case S4-LGPL	GPL	69.5% ●	24.9%	5.6%
	LGPL	96% ●	1.9%	2.1%
	MPL	18.4%	70.5% ●	11.1%
Case S4-MPL	GPL	53.5%	31% ●	15.5%
	LGPL	54.4%	29.2% ●	16.4%
	MPL	90.3% ●	2.1%	7.5%
Case S5 - Removed				
Case S6-GPL	GPL	74.2% ●	24.2%	1.6%
	LGPL	6.5%	90% ●	3.5%
	MPL	6.2%	86.8% ●	7%
Case S6-LGPL	GPL	57.4% ●	36.4%	6.2%
	LGPL	80.6% ●	16.9%	2.4%
	MPL	14.1%	77.5% ●	8.4%
Case S6-MPL	GPL	43.2% ●	42.9%	13.9%
	LGPL	44.4% ●	41.5%	14.1%
	MPL	80.1% ●	11.1%	8.9%
Case S7	GPL	2.1%	92.5% ●	5.3%
	LGPL	2.7%	92% ●	5.3%
	MPL	4%	88.5% ●	7.5%

Table 4.4: 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

tions 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 4.6 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.

Case Comments

We also analyzed 462 comments provided by participants for the 17 cases of interest. Table 4.5 describes the seven codes that resulted from the process described in Chapter 4.1.2; these codes include comments 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 4.7 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 `U`nsure 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

Table 4.5: 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.

Table 4.6: 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

in the assumption coding, such as which licenses could subsume other licenses, when code could be dual licensed and when code could be re-licensed. This data helps reinforce *Observation 4*.

4.2.4 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 4.1.

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.

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"*.

4.3 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 [20]. 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 4.2, 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.

4.4 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*).

Table 4.7: 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

Chapter 5

The Context of License Incompatibilities

The survey results indicate that software developers struggle with license incompatibilities and that technical details and license interaction seem to be particularly important. The survey considered hypothetical cases, with limited information about real cases gathered through open-ended comments by participants.

To gain insight into the context in which license incompatibilities arise, I posed two research questions:

RQ#1 In what real-life situations do license incompatibilities arise?

RQ#2 How do software developers reason about and handle license incompatibilities?

To investigate the first question, I mined and analyzed questions posed by developers, and the answers they received, on popular online question-and-answer communities. To investigate both questions, I also interviewed software developers in industry to better understand how software developers approach and handle license incompatibility issues in practice.

5.1 Mining Real-World Cases

To obtain real-world cases involving software licensing issues, I mined information from StackExchange, a popular network of question-and-answer (Q&A) communities. One community within StackExchange is Open Source Stack Exchange, which is a community for ”people organizing, marketing or licensing open source development projects”[1]. In addition to this source, I considered two other StackExchange communities —Software Engineering and Stack Overflow—because these are popular communities that also deal with licensing issues and may be more likely to see a mix of open and closed source licensing questions.

To find real-world cases on these sites, I performed queries about software licenses. To focus the exploration, I narrowed my queries to those involving two of the three licenses considered in the survey: GPL, LGPL and MPL. I made this choice to increase the likelihood of finding discussions of license incompatibilities, because more than one license is mentioned, and to follow-up on situations hinted at in the survey. I performed three queries on each of the three StackExchange communities to search for questions containing the following combinations in their body: ”GPL” and ”LGPL”, ”GPL” and ”MPL”, or ”LGPL” and ”MPL”. The results of these queries are exchanges from the community that consist of a question with different answers and threaded replies to each answer.

Performing the queries on the StackExchange communities produced 60 results. Table 5.1 shows the number of questions retrieved from the different communities for each search query. As analysis of each result is qualitative, I chose to sample from the results to analyze whether a license incompatibility had occurred. I randomly sampled a third of the results to analyze. For each selected result, I summarized the question and tagged the question with codes from the qualitative analysis of the survey’s case comments (Table 4.5). I also considered whether any new codes needed to be added to capture the intent of the question; no new codes emerged. Table 5.2 reports on the number of times each code was assigned to one of the questions. A full list of the questions analyzed, their summary and assigned codes can be found in Appendix A.

As Table 5.2 shows, seven of the 20 questions analyzed were assigned the

[1] https://opensource.stackexchange.com/

Table 5.1: Number of questions per community for each query

Community	GPL and LGPL	GPL and MPL	LGPL and MPL
Stack Overflow	30	4	3
Software Engineering	15	0	4
Open Source	4	0	0

Table 5.2: Codes used to describe real-world cases

Code	# Questions	Description
Invalid (*I*)	7	Unrelated to licensing.
License Interaction (*LI*)	9	Actions possible with more than one license.
Specific Case (*SC*)	2	Relicensing or dual license.
Technical Detail (*TD*)	10	Technical aspects of case.

invalid code (`I`), meaning that the question was not related to licensing issues. These questions are not considered in any further analysis and can be seen as false positives due to the fact that our search queries considered any questions containing two of the three open source licenses in their body.

For each of the analyzed questions assigned a valid code, Table 5.3 shows the exact codes assigned to the question. From the analysis of the remaining 13 questions, three kinds of contextual situations emerged. I describe each in turn.

Situation #1: Constrained Software Evolution

Many of the cases analyzed involved developers looking for components released under permissive licenses because they were unable to use open source components under more restrictive licenses such as GPL and LGPL. The authors had a specific task they were trying to complete and were looking for existing open source components that implemented the required functionality. This theme occurred in eight cases (Questions 1, 3, 5, 6, 8, 10, 11, and 13).

For example, the author of question #6 was asking for a recommendation of an asymmetric cryptographic algorithm. The author was willing to implement the algorithm from scratch or use an existing library that implements the algorithm, but the library could not be licensed under GPL or LGPL because the project uses static linking, and the author did not want to have GPL or LGPL apply to their

Table 5.3: Codes assigned to each question

#	Community	Code(s)
0	Software Engineering	*SC, TD*
1	Stack Overflow	*SC*
3	Stack Overflow	*TD*
4	Software Engineering	*TD, LI*
5	Stack Overflow	*TD*
6	Stack Overflow	*TD*
7	Stack Overflow	*LI*
8	Stack Overflow	*TD, LI*
10	Stack Overflow	*TD, LI*
11	Software Engineering	*LI, TD*
13	Open Source	*LI, TD*
15	Software Engineering	*TD, LI*
19	Stack Overflow	*LI, TD*

entire project. Similarly, the author of question #5 was trying to improve the way a legacy application encodes/decodes video, but seemed to be constrained because the current solution was built around an existing open source library (`ffmpeg`). The author recognizes that `ffmpeg` is a popular solution to the problem at hand, but is looking for an alternative because a co-worker made him believe that changes or improvements to the existing code would result in a violation of `ffmpeg`'s licensing requirements.

Situation #2: Technical Decisions Impact Future Licensing

Another theme in the analyzed question was that of a developer who is struggling with licensing issues because of previous technical decisions. Some of these were due to characteristics of the project itself. This theme occurred in six question (Questions 3, 5, 11, 13, 15, and 19).

For instance, the author of question #3 is in need of a library for an Android application, but states that libraries under GPL or LGPL cannot be used. One of the respondents (*OleGG*) seems to think that the author holds a misconception about LGPL and suggests that it is possible to use LGPL components without making the author's code open source. Another respondent (*bovine*) replies and explains that

OleGG's suggestion only applies when there is a "suitable shared library mechanism", which is not the case for Android projects by definiton because the resulting application would be released as a single *Dalvik Executable* (DEX) file.

In a different question (#11), a developer is facing difficulties while trying to eliminate dependencies acquired during the development of a prototype. The team had decided to use a number of open source components and made design decisions based on the existence of these components. Later on, they realized that one of the components imposed licensing requirements that were incompatible with their intended use, requiring a significant amount of effort to work around previous design decisions and eliminate that dependency.

Situation #3: Developers Struggle to Find Appropriate Licenses

In most of the cases analyzed, authors described facing difficulties related to licensing while working with existing open source components. In a few cases, in contrast, authors describe trying to choose a license for their own projects. They seem aware of the implications of licensing decisions and are trying to better understand their options. Questions #0 and #4 are examples of this theme. The author of the first question seems to be aware of the implications of a number of popular licenses, but is unable to identify which licensing option meets a number of requirements. Similarly, the author of the second question needs help to confirm his understanding of how MPL and LGPL relate to the specific details of a project. The author wants to make sure he chooses a license that effectively imposes certain restrictions on those making use of his open source component in the future.

5.2 Interviews

To better understand the context in which license incompatibility issues take place and how developers approach and handle these issues in practice, I conducted semi-structured interviews with developers in industry. Appendix B contains the script for these interviews.

To find participants who are more likely to have experienced license incompatibility issues in practice, I recruited software developers that have built or are currently building on open source software in their projects. Six developers vol-

Table 5.4: Job title and company size for each participant

P#	Job Title	Company Size
P1	Software Developer (DevOps)	51 to 200
P2	Senior Software Engineer	501 to 1000
P3	Solutions Architect	51 to 200
P4	Principal Engineer	201 to 500
P5	Principal Engineer	201 to 500
P6	Software Engineer (DevOps)	201 to 500

unteered to participate and were interviewed via a web conferencing system. Table 5.4 shows, for each participant, their job title and company's size. All participants are male. The interviews were recorded and simultaneously analyzed by an experienced researcher in the field and the author. We then compared and discussed our findings to identify themes that might shed light on how developers handle license incompatibility situations in practice. I describe each one of the themes in turn.

Theme #1: Developers and software development companies are cautious of restrictive licenses. Five interviewees mentioned avoiding software components under restrictive licenses, such as GPL. They also mentioned that, in such situations, they tend to consider alternative components available under permissive licenses.

Two interviewees mentioned situations in which GPL components were rewritten as the functionality was required, but GPL components could not be used. An interviewee also mentioned that, in his organization, there can be requests to use components under the LGPL license; such a request would be made if the component could be used in the right technical structure or if the component was the best in its class.

Theme #2: Developers use the technical architecture of the product to enable the use of components under restrictive licenses. Three interviewees talked about using the architecture of the system to enable the use of components under restrictive licenses. One of them mentioned a situation in which a GPL licensed component was used in development, but not in the version of the product shipped to customers.

Similarly, other participants talked about how specific architectural decisions can enable the use of components under restrictive licenses, e.g., by offering software as a service or having the end-user download a specific component.

Theme #3: Developers assume communities use a certain kind of license. Three interviewees mentioned that they believe certain open source communities would generally use permissive licenses. One of the interviewees seemed to believe that JavaScript packages found on *npm* are usually available under permissive licenses. While talking about a Java project on which he had been working, another participant admitted to making what is a dangerous assumption in his opinion: namely that components available through Maven are under permissive licenses. Finally, a third participant, as a member of an open source community, suggested that most members of that community use a specific license.

Theme #4: License incompatibility situations arise but not frequently. Five interviewees mentioned remembering a situation in which license incompatibility was a concern, but such situations do not seem to happen often for these individuals. One interviewee mentioned the frequency of about once per year dealing with a license incompatibility between components.

Theme #5: Developers use specialized tools to detect licenses. Three interviewees mentioned having used specialized tools to identify what licenses are used by a project and its dependencies and help them detect possible licensing issues.

5.3 Threats to Validity

The extent to which results of the real-world cases describe the frequency of situations in which license compatibilities arise is unclear for two reasons.

First, the cases I analyzed were obtained from a single source (StackExchange) and might not be representative of all kinds of licensing incompatibility issues faced by software developers in practice. I focused on StackExchange instead of other online communities dedicated to specific licenses because: 1) StackExchange communities are widely used by software developers,StackOverflow alone has more than 14 million questions and is visited by 51,000 developers per month on average.[2] and 2) StackExchange draws a wide variety of developers given its

[2]https://stackoverflow.com/company

diverse forums.

Second, I obtained the real-world cases using search queries that included the names of the three licenses used in the survey. I made this decision because the purpose of this analysis is to better understand the context in which licensing issues first identified in the survey happen. Just as the results of the survey are not necessarily representative of all kinds of licensing issues developers struggle with, the contextual situations identified in the real-world cases I analyzed cannot be generalized to all other licenses. A better understanding of the types of existing open source licenses and of their different characteristics would help us understand to what extent the contextual situations I identified can be generalized.

The interviews with software developers also may not represent all themes that arise in license incompatibility scenarios. The population of developers was comprised of developers motivated to discuss issues involving open source licenses. They came from organizations that had between 51 and 1000 employees. Licensing incompatibility issues might be more or less frequent in organizations of different sizes and the themes that emerged could be different given a different population.

5.4 Summary of Results

The analysis of real-world cases suggests that the contextual situations in which license incompatibility takes place involve a mix of technical details and license interactions. In many of these cases, technical decisions had an impact on future licensing issues, while in other cases licensing issues imposed constraints on software evolution. In a few of the cases developers were struggling to understand what licensing approach to use in order to impose the restrictions they intended for their own open source projects.

The developers I interviewed are cautious of restrictive licenses, such as GPL and LGPL. They use specialized tools to detect the licenses being used in their projects and possible licensing issues. When faced with a situation in which a component that suits their needs is under a restrictive license, developers tend to rely on open source communities to find alternative components that can be used. They sometimes leverage the product's technical architecture to enable the use of components under restrictive licenses and might rewrite the required functionality if necessary.

Chapter 6

Conclusion and Future Work

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 (MPL, GPL, LGPL) 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.

The analysis of real-world cases and the interviews with experienced software developers in industry shed light on the context in which license incompatibility issues take place in practice.

We found that although some developers make use of specialized tools to identify licenses being used in a project and potential problems arising from the interaction of those licenses, others rely on licensing guidelines provided by their organization to decide if a component under a license can be used.

Across the cases we analyzed and interviews we conducted, there is a clear tendency towards avoiding restrictive licenses in favour of more permissive licenses. This tendency can be seen in many real-world cases in which developers ask for help to find a component under a permissive license and that satisfies their needs as well as from the experiences of the developers I interviewed and the guidelines adopted at their organizations.

In situations where a specific component seems to fit the developers' needs but is incompatible with their project's licensing guidelines, the software developers interviewed and the cases analyzed suggest at least three approaches to resolve this issue.

The first approach is to **find an alternative component available under a permissive license.** The increasing number of open source components available to developers allows them to look for alternative components released under a license compatible with their projects. In this case, developers rely on the open source community and seemed fairly confident that many of the components available through existing package management tools, such as npm and Maven, are under permissive licenses.

The second approach is to **rewrite a component from scratch.** Depending on the effort necessary to implement the required functionality, developers may prefer to rewrite the component themselves and avoid any risks involved in using components under restrictive licenses.

The third approach is to **restructure the system code to enable the use of a component.** In some cases, developers can enable the use of certain components or work around them by restructuring the way that component is being used in their project. This was observed in multiple interviews and the examples given involved changes to the packaging and distribution of the software (e.g., by adopting a software as a service (SaaS) distribution model) and changes to the architecture of the product (e.g., removing a component from a software product and allowing the end-user to download such component).

To improve the support for developers in dealing with license incompatibilities, enhanced tool support is needed. This enhanced support can build on existing efforts of researchers. For example, Germán and Hassan [7] provide a model for identifying possible mismatches when different open source licenses interact, but this model is not able to recognize code structures that cause these mismatches. Vendome's research goes further, suggesting a need for tools that are able 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 [18]. The comments we analyzed from participants completing our survey and the analysis of the context in which license incompatibility happens suggest that such a recommendation engine may need to be more extensive and robust than suggested by Vendome. Our results indicate a need for tools capable of analyzing the structure of the code and the technical details of the project while taking into account the exact requirements imposed by the licenses involved in that project. Such tools should be able to suggest ways to restructure the 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 [3], may provide a starting point for building such tools.

Bibliography

[1] On the continuing decline of the gpl. http://blogs.the451group.com/opensource/2011/12/15/on-the-continuing-decline-of-the-gpl/, 2011. → pages 5

[2] D. A. Almeida, G. C. Murphy, G. Wilson, and M. Hoye. Do software developers understand open source licenses? In *Proceedings of the 25th International Conference on Program Comprehension*, ICPC '17, pages 1–11. IEEE Press, 2017. → pages v

[3] 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*, 11(11):730, 2010. → pages 6, 37

[4] S. Clarke. Measuring API usability. *Dr. Dobb's Journal, Special Windows/NET Supplement*, 2004. → pages 1

[5] J. Corbin and A. Strauss. Grounded theory research: Procedures, canons and evaluation criteria. *Qualitative Sociology*, 13:3–21, 1990. → pages 12

[6] 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*, ICSE '10, pages 145–154. ACM, 2010. → pages 5

[7] D. M. German and A. E. Hassan. License integration patterns: Addressing license mismatches in component-based development. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 188–198. IEEE Computer Society, 2009. → pages 6, 37

[8] D. M. German, M. Di Penta, and J. Davies. Understanding and auditing the licensing of open source software distributions. In *Proceedings of the 2010 IEEE 18th International Conference on Program Comprehension*, ICPC '10, pages 84–93. IEEE Computer Society, 2010. → pages 2

[9] D. M. German, Y. Manabe, and K. Inoue. A sentence-matching method for automatic license identification of source code files. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pages 437–446. ACM, 2010. → pages 6

[10] 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*, MSR '11, pages 63–72. ACM, 2011. → pages 2

[11] 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*, pages 245–256. Springer, 2013. → pages 5

[12] M. P. Robillard and R. Deline. A field study of api learning obstacles. *Empirical Software Engineering*, 16(6):703–732, Dec. 2011. → pages 1

[13] M. B. Rosson and J. M. Carroll. The reuse of uses in smalltalk programming. *ACM Trans. Comput.-Hum. Interact.*, 3(3):219–253, Sept. 1996. → pages 1

[14] R. Sen, C. Subramaniam, and M. Nelson. Determinants of the choice of open source software license. *J. Manage. Inf. Syst.*, 25(3):207–240, Dec. 2008. → pages 5

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

[16] 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*, 17(2): 126–144, June 2006. → pages 5

[17] C. Vendome. A large scale study of license usage on github. In *Proceedings of the 37th International Conference on Software Engineering - Volume 2*, ICSE '15, pages 772–774. IEEE Press, 2015. → pages 1

[18] C. Vendome and D. Poshyvanyk. Assisting developers with license compliance. In *Proceedings of the 38th International Conference on Software Engineering Companion*, ICSE '16, pages 811–814. ACM, 2016. → pages 37

[19] 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*, ICPC '15, pages 218–228. IEEE Press, 2015. → pages 6

[20] C. Vendome, M. Linares-Vasquez, G. Bavota, M. Di Penta, D. M. German, and D. Poshyvanyk. When and why developers adopt and change software licenses. In *Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, ICSME '15, pages 31–40. IEEE Computer Society, 2015. → pages 2, 6, 7, 23

[21] C. Vendome, M. Linares-Vásquez, G. Bavota, M. Di Penta, D. German, and D. Poshyvanyk. Machine learning-based detection of open source license exceptions. In *Proceedings of the 39th International Conference on Software Engineering*, ICSE '17, pages 118–129. IEEE Press, 2017. → pages 6

Appendix A

StackExchange Cases

Question #0: Help me choose an Open-Source license
Link: https://softwareengineering.stackexchange.com/questions/50764/help-me-choose-an-open-source-license
Codes: *Specific Case, Technical Details*
Summary: Question asks for help to select a license that fulfills a number of requirements. The author is implementing a library and wants to allow its use in other open source projects regardless of their licenses, but not in closed-source/commercial projects unless s/he grants written authorization. The proposed solution is dual licensing and answers suggest it isn't possible to fulfill all of the requirements.

Question #1: digital signature with specific itext version 4.2.0
Link: https://stackoverflow.com/questions/25879897/digital-signature-with-specific-itext-version-4-2-0
Codes: *Specific Case*
Summary: The author wants to know if the iText library offers a certain feature in version 4.2 (instead of 5.0) because the latest version has been released under AGPL (according to its GitHub repository, they "moved to the AGPL to improve their ability to sell commercial licenses"). Version 4.2 was released under dual licensing (MPL/LGPL).

Question #2: Use default authentication and separate cloaking/impersonation in

DCOM call
Link: https://stackoverflow.com/questions/1993651/use-default-authentication-and-separate-cloaking-impersonation-in-dcom-call
Codes: *Invalid*
Summary: Author has a programming question and mentions that his project will be released under GPL/MPL as an incentive (”So your help will go GPL/MPL.”).

Question #3: Use default authentication and separate cloaking/impersonation in DCOM call
Link: https://stackoverflow.com/questions/9213152/override-mapview-tile-source
Codes: *Technical Details*
Summary: Author is looking for an Android component/library that overrides the MapView behaviour. The component cannot be based on GPL/LGPL. One of the answers suggests a couple of libraries that are under LGPL, but the author reiterates that he was trying to avoid anything under LGPL. The respondent clarifies to the author that it's possible to use LGPL libraries without making his code available. A second respondent goes further and explains that it's possible to use LGPL if there's a ”suitable shared library mechanism”, which is not the case for Android because the resulting application will be a single DEX (Dalvik Executable format) file.

Question #4: Mozilla Public License (MPL 2.0) vs Lesser GNU General Public License (LGPL 3.0)
Link: https://softwareengineering.stackexchange.com/questions/221365/mozilla-public-license-mpl-2-0-vs-lesser-gnu-general-public-license-lgpl-3-0
Codes: *Technical Details, License Interaction*
Summary: The author wants to release a library as open source while making it possible for others to use it in proprietary/closed projects. He is considering either LGPL or MPL and identifies the type of linking as the main difference (static for MPL and dynamic for LGPL). He believes that LGPL imposes extra obstacles for packaging and that neither of them (LGPL or MPL) requires extensions of the software library to be released as open-source. His solution is: ”With the help of

the discussion in the accepted answer, I choose to stick to the MPL because of the popularity, freedom in linking and because it is an official, unmodified license."

Question #5: How to encode/decode video using C#?
Link: https://stackoverflow.com/questions/1834667/how-to-encode-decode-video-using-c
Codes: *Technical Details*
Summary: Author is trying to find a better way of encoding/decoding video to improve a legacy application built around ffmpeg, an open-source library released under GPL or LGPL. The author seems to be confused about what he can or cannot do with the current code (co-workers made him believe that improvements/changes to the current code might result in violations of ffmpeg's license and possibly its dependencies' licenses as well.

Question #6: Fast asymmetric cypher for C++ application
Link: https://stackoverflow.com/questions/2247697/fast-asymmetric-cypher-for-c-application
Codes: *Technical Details*
Summary: Author is looking for an algorithm for asymmetric cryptography so that he can implement it himself OR a library that implements the algorithm. He needs to use static linking in his proprietary application, which means that the library cannot be under GPL or LGPL. He says that MIT/BSD or public domain code would be fine.

Question #7: GPL, LGPL licensed product in a comercial software [closed]
Link: https://stackoverflow.com/questions/12030426/gpl-lgpl-licensed-product-in-a-comercial-software
Codes: *License Interaction*
Summary: Author is confused about what GPL, LGPL allows him to do. He wants to use a piece of software's documentation in his commercial product. It turns out that the product (processing.org) is released under GPL or LGPL, but its documentation is under Creative Commons.

Question #8: How to handle DWG files in C++

Link: https://stackoverflow.com/questions/22597111/how-to-handle-dwg-files-in-c
Codes: *Technical Details, License Interaction*
Summary: Author asks for a library to handle a certain file type but specifies that it has to be under a permissive license (e.g, MIT or BSD) or a weak copyleft license that allows him to statically link (he mentions LGPL but isn't sure if it allows static linking).

Question #9: Restful Webservice Java, server side
Link: https://stackoverflow.com/questions/14971059/restful-webservice-java-server-side
Codes: *Invalid*
Summary: Author is looking for a simple embedded servlet container and says it cannot be under GPL/LGPL.

Question #10: iText 2.1.7 in commercial Project
Link: https://stackoverflow.com/questions/8705697/itext-2-1-7-in-commercial-project
Codes: *Technical Details, License Interaction*
Summary: Author wants to use a library (iText, a Java PDF library) in his commercial project but isn't sure about how to proceed and what is allowed. The library used to be under MPL/LGPL until version Version 2.1.7, but is now under AGPL. He has questions regarding a number of different aspects, such as: a) if the lib can be used in a commercial project;

b) if he can choose which license to use (MPL or LGPL);

c) if he has to ship only the license text or the library source code as well;

d) if he can put everything in an executable or if the iText should be in a separate jar file.

Question #11: Legal problem on lifting from open source libraries
Link: https://softwareengineering.stackexchange.com/questions/321722/legal-problem-on-lifting-from-open-source-libraries
Codes: *License Interaction, Technical Details*

Summary: Author used a number of libraries during prototyping development and now wants to reduce external dependencies. It turns out that he is heavily using a class (LazyInitializer) from the apache common-lang3 library (under Apache 2.0) to implement the Singleton Pattern and isn't sure about the licensing implications. He says that removing that dependency would implicate in the creation of more than 40 static inner private classes to implement the Singleton Pattern. Finally, he simply asks what are the implications of other major open source licenses (mentions MIT, GPL, LGPL, and BSD).

Question #12: Is there any free opensource (GNU GPL, LGPL etc) Comet Video Chat? (PHP)
Link: https://stackoverflow.com/questions/2766018/is-there-any-free-opensource-gnu-gpl-lgpl-etc-comet-video-chat-php
Codes: *Invalid*
Summary: Author is looking for an open-source/free Comet Video Chat implementation to use in his own CMS (Content Management System).

Question #13: Which licenses can I use?
Link: https://opensource.stackexchange.com/questions/4484/which-licenses-can-i-use
Codes: *License Interaction, Technical Details*
Summary: Author understands that some licenses are not compatible with each other and is concerned with compatibility issues between GPL or LGPL and a license already in use in his project (MongoDB driver in Java released under AGPL 3). He wants to know which licenses are compatible if the project or libs in the project are under GPL, LGPL, and AGPL. Respondent explains AGPL license and possible license combinations. Another respondent suggests that LGPL and AGPL code are compatible provided that they're linked (doesn't specify if dynamic or static linking though).

Question #14: Qt License now after Digia bought it
Link: https://stackoverflow.com/questions/11893121/qt-license-now-after-digia-bought-it
Codes: *Invalid*

Summary: Author doesn't know if a set of libraries (Qt platform) is still under the same license (LGPL) and is afraid of any changes that might prevent its use in commercial projects. Respondent quotes the company's website suggesting that both versions (one under LGPL and the other under a commercial license) will continue to be maintained.

Question #15: Can I use GPL, LGPL, MPL licensed packages with my application and make it closed source?
Link: https://softwareengineering.stackexchange.com/questions/125606/can-i-use-gpl-lgpl-mpl-licensed-packages-with-my-application-and-make-it-close
Codes: *Technical Details, License Interaction*
Summary: The author is not sure about the extent to which a program released under GPL affects the licensing of other software in the same "aggregate" [2] (a bundle of individual programs under different licenses). In this case, he mentions a company building closed-source applications mixed with/using BusyBox [1], an application released under GPL.

The most accepted answer suggests that GPL allows the use of open and closed-source software in an aggregate as long as the GPL license is respected, i.e., the source code of the software under GPL is released. According to the respondent, GPL doesn't extend to other programs running on the same computer. It's possible to have closed-source software interacting/communicating with open-source code under GPL. "As a rule of thumb, the GPL reaches as far as the address space of the licensed code."
Question #16: Efficient Multiply/Divide of two 128-bit Integers on x86 (no 64-bit)
Link: https://stackoverflow.com/questions/8776073/efficient-multiply-divide-of-two-128-bit-integers-on-x86-no-64-bit
Codes: *Invalid*
Summary: Author wants to know how to efficiently do something and cannot use GPL/LGPL code.

Question #17: How to read utf-16 file into utf-8 std::string line by line
Link: https://stackoverflow.com/questions/29012472/how-to-read-utf-16-file-into-utf-8-stdstring-line-by-line

Codes: *Invalid*
Summary: Author wants to know how to do something and cannot use GPL/LGPL code.
Question #18: Which Licenses to ship within an NW.js - application?
Link: https://stackoverflow.com/questions/29234304/which-licenses-to-ship-within-an-nw-js-application
Codes: *Invalid*
Summary: Author is using a library built on top of Chromium, which has a number of dependencies under different licenses. The author doesn't know what licensing information he has to provide to respect those dependencies' licenses.

Question #19: Requirements for using ffmpeg to create mpeg4 files in SaaS solution
Link: https://stackoverflow.com/questions/39082573/requirements-for-using-ffmpeg-to-create-mpeg4-files-in-saas-solution
Codes: *License Interaction, Technical Details*
Summary: Author is building a SaaS solution and is trying to avoid a commonly used library (ffmpeg) because it introduces ”licensing hell” not only with GPL/L-GPL, but because it is not available under a commercial license. He doesn't not how licensing applies to SaaS solutions since the code will not be distributed.

A respondent clarifies that a few of the license's requirements don't seem to apply to his case (SaaS solution) based on his own interaction with the company that owns the library's rights.

Appendix B

Interview Script

1. What is your current role in your organization?
2. What are your main responsibilities?
3. How are you involved with the software products produced at your organization?
    (a) Do you write code? Select open source components? Test?
4. How are licensing decisions made at your organization?
    (a) How are developers involved? Could you give a few examples of decisions you have been involved in?
5. Have you ever encountered license incompatibility issues?
6. Could you describe a particular scenario in which license incompatibilities occurred?
    (a) What were the licenses? What was the structure of the code carrying the licenses?
    (b) How did the code with different licenses interact?
    (c) What was your role in identifying or resolving the challenge?
7. How often has a situation like this occurred?