Exploring Software Licensing Issues Faced by Legal Practitioners

Nathan James Wintersgill

Monroeville, Pennsylvania

Bachelor of Arts, Bucknell University, 2020

A Thesis presented to the Graduate Faculty
of The College of William and Mary in Virginia in Candidacy for the Degree of
Master of Science

Department of Computer Science

College of William and Mary in Virginia
January 2024

APPROVAL PAGE

This Thesis is submitted in partial fulfillment of
the requirements for the degree of

Master of Science

[illegible]

Nathan Wintersgill

Approved by the Committee, November 2023

[illegible]

Committee Co-Chair
Denys Poshyvanyk, Chancellor Professor, Computer Science
College of William & Mary

[illegible]

Committee Co-Chair
Oscar Chaparro, Assistant Professor, Computer Science
College of William & Mary

Adwait

Adwait Nadkarni, Associate Professor, Computer Science
College of William & Mary

COMPLIANCE PAGE

Research approved by

Protection of Human Subjects Committee

Protocol number(s): PHSC-2023-05-16-16375

Date(s) of approval: 07/16/2023

ABSTRACT

Most modern software products incorporate open source components, which requires compliance with each component's licenses. As noncompliance can lead to significant repercussions, organizations often seek advice from legal practitioners to maintain license compliance, address licensing issues, and manage the risks of noncompliance. While legal practitioners play a critical role in the process, little is known in the software engineering community about their experiences within the open source license compliance ecosystem. To fill this knowledge gap, a joint team of software engineering and legal researchers designed and conducted a survey with 30 legal practitioners and related occupations and then held 16 follow-up interviews. We identified different aspects of OSS license compliance from the perspective of legal practitioners, resulting in 18 key findings in three main areas of interest: the general ecosystem of compliance, the specific compliance practices of legal practitioners, and the challenges that legal practitioners face. We discuss the implications of our findings.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my co-advisors, Dr. Denys Poshyvanyk and Dr. Oscar Chaparro, for continually providing their careful guidance and patient input throughout this investigation.

I also wish to thank my colleagues Trevor Stalnaker, Laura A. Heymann, Sang Hwan Lee, and JT Shorten for bringing their invaluable perspectives and insights, enriching this research, and ultimately facilitating the completion of this thesis.

Thank you to my committee members for offering your time, consideration, and judgement at this critical stage in my research.

Thank you to the survey and interview participants who graciously donated their time to share their valuable insights and expertise, without whom this investigation could not have come to fruition.

To Ruth Cain, who learned to believe in me before I did.

LIST OF TABLES

LIST OF FIGURES

Exploring Software Licensing Issues Faced by Legal Practitioners

Chapter 1

Introduction

Over more than two decades, the open source software (OSS) community has propelled the software industry forward, creating a dynamic supply chain where software systems are often built by integrating existing OSS components, such as libraries and frameworks [23, 43, 32, 29]. This reuse of components allows developers to forgo reinventing the wheel by freely utilizing existing solutions to common problems and thus accomplish their tasks more productively. To promote open collaboration and support the software supply chain, OSS components are typically released under one or more OSS licenses, which define the terms governing the components' distribution, modification, and reuse. As software is protected by copyright and patent laws [19], software projects integrating such components must comply with the terms of any applicable licenses, and failing to do so can result in significant legal [33, 79], reputational [34, 46], and financial consequences [75, 44] for developers and organizations.

Although ensuring software license compliance is thus a critical process for developers and organizations, the process is often challenging [42, 39, 26, 77]. Software systems often integrate hundreds or even thousands of OSS components, distributed under one or more licenses [29]. These licenses can conflict with one another, as there are hundreds of OSS licenses with different (in)compatibility levels [14, 11]. Additionally, licenses are written in legal terms that can be subject to different interpretations, and developers often struggle to accurately apply these terms [26, 39, 77].

Given the risks of license noncompliance, organizations often seek legal advice from inhouse or outside counsel on license compliance issues, guidance on addressing such issues, and strategies for maintaining compliance. While legal practitioners play a critical role in the process of license compliance, little is known in the software engineering (SE) community about their perception and experiences within the OSS license compliance ecosystem, including their methodologies for assisting organizations during license compliance and the challenges they face in this process. Prior studies investigated OSS license compliance issues and their impact mostly from a developer/user perspective [26, 76, 77, 80, 66], rather than from a legal perspective.

To fill this knowledge gap in the SE community, we conducted a qualitative study that examined the experiences of a group of legal practitioners specializing in OSS license compliance in the US. The study, conducted by a joint team of SE and legal researchers, surveyed 30 legal professionals and compliance experts, who answered an online survey that probed into various aspects of their work and past experiences. We subsequently conducted interviews with 16 of the respondents to delve deeper into their experience within the OSS license compliance ecosystem.

The study yielded insights into OSS license compliance as viewed through the lens of legal practitioners. These findings encompass various dimensions, including license selection, creation, proliferation, evolution, and interpretation; license enforcement; how licensing issues are resolved; risk management strategies; tool usage for license compliance; and prevention strategies, including training, education, and communication with developers. We qualitatively analyze these findings and discuss opportunities for improvement as well as avenues for future research.

In summary, the main contributions of this thesis are: (1) a rigorous assessment of the current state of the OSS license compliance ecosystem from the perspectives of 30 legal practitioners and compliance experts; (2) an in-depth analysis of how these individuals perform license compliance tasks and the associated challenges they face; and (3) a thorough discussion of the findings and their implications, highlighting avenues for future work. We provide a replication package with additional data for verifiability [64].

Chapter 2

Background

Software, including OSS, is protected by copyright law in the United States as a general matter [19]. The owner of copyright in a work has several rights under U.S. copyright law, including the right to reproduce the work, to create derivative works, and to distribute the work, as well as the right to authorize others to engage in these activities [20]. Although the copyright in a work can be sold or transferred, a copyright license is the mechanism by which a copyright owner retains copyright in the work while authorizing another party to use the work in ways that would otherwise constitute infringement, sometimes subject to stated conditions. [59].

The developer of OSS typically chooses an existing OSS license under which to issue their work rather than creating a new license. At present there are 114 licenses recognized by OSI [11] and 526 by SPDX [14], with a handful, such as MIT, GPL, and Apache, being the most prevalent [31]. OSS licenses generally fall into one of two categories: permissive and restrictive (also called copyleft). Permissive licenses (such as MIT [10] and BSD [1]) typically impose few restrictions and requirements on those using a piece of software (*e.g.*, the requirement to provide notice files containing OSS component attribution or licenses), whereas copyleft licenses (GPL [7], LGPL [8], *etc.*) typically require that the full source code of the resulting work be made available and that the project is licensed under the same license (leading some to characterize such licenses as "viral").

Because software licenses contain conditions on use, entities that use OSS components in their own development projects must ensure that such use complies with the requirements of the license to avoid a claim of copyright infringement. Prior studies have shown that this task can be challenging for developers to complete on their own [77]. Companies often hire legal counsel (in-house or outside) to assist with compliance tasks and to navigate the myriad of potential licensing issues [77]. These professionals provide their clients with guidance on license interpretation, create educational resources for developers, and help clients manage the risks associated with using OSS.

OSS licensing can involve several areas of law beyond copyright law, such as patent law, trademark law, cybersecurity, and privacy law. We focus in this thesis on U.S. copyright law and legal practitioners located in the U.S., although their work may involve jurisdictions around the world.

Chapter 3

Methodology

This study, which involves a collaboration between SE and law researchers, aims to analyze OSS license compliance issues from the legal practitioner perspective through surveys and follow-up interviews. We address the following research questions (RQs):

RQ$_1$: *What is the ecosystem of license compliance experienced by legal practitioners?*

RQ$_2$: *How do legal practitioners perform license compliance in this ecosystem?*

RQ$_3$: *What challenges do legal practitioners face during license compliance?*

In **RQ1**, we investigate how OSS licensing compliance *generally* works from the experience of legal practitioners, including how OSS licenses are selected and used, the kinds of violations that occur, how licenses are enforced, and how disputes are resolved. **RQ2** aims to investigate *specific* compliance activities performed by practitioners, including when compliance is performed, who is involved in the process, how risk is managed, how education/training is performed, and how tooling is used. **RQ3** investigates *challenges* faced by legal practitioners during the process.

The study combined an online survey and follow-up interviews conducted with legal practitioners and others specializing in OSS licensing within the U.S. We used various strategies to find potential participants and used an open-coding methodology to analyze survey and interview responses. This section details the research methodology (see Figure 3.3), Chapter 4 presents the study results and their analysis, and Chapter 7 discusses the implications of our findings. The methodology was approved by The College of William & Mary's Protection of Human Subjects Committee.

An overview of our approach can be seen in Figure 3.1.

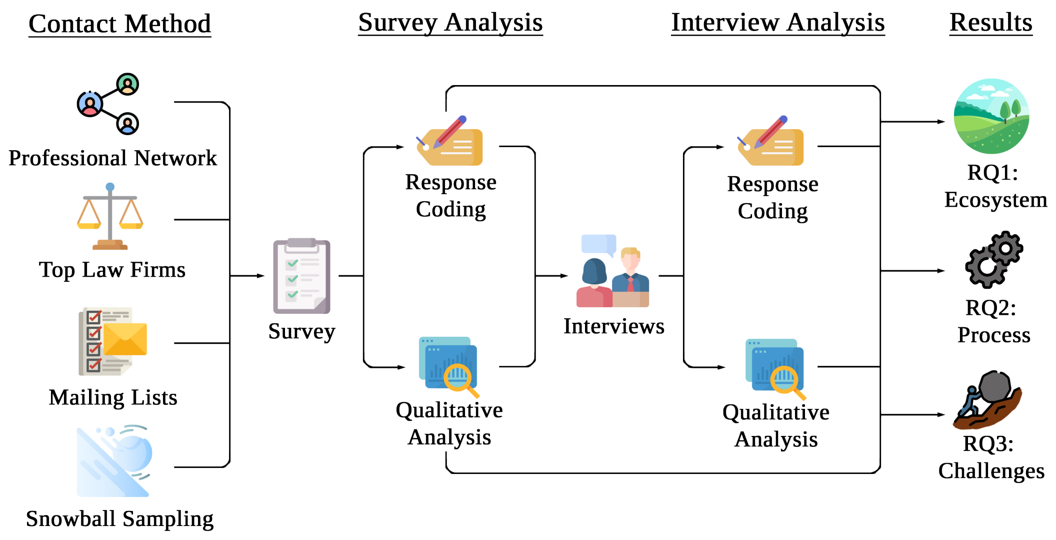

Figure 3.1: Research methodology (image credits in Appendix B)

3.1 Survey design and participant identification

In designing our survey, we followed general guidelines for survey design [45] as well as SE-specific guidelines [67, 50, 51, 52, 53, 54]. The survey questionnaire went through multiple iterations with reviews and modifications from four SE and three law researchers, making sure that questions were written clearly and concisely to avoid confusion and bias. The survey was designed to be completed in about 15 minutes to be mindful of participants' time and to maximize the number of useful responses. The survey included mostly open-ended questions and asked about client practices, past experiences, edge cases (*e.g.*, multi-licensing), tooling, and information needs during OSS license compliance. Respondents were asked to self-identify as in-house or outside counsel or as other roles.

Our survey targeted legal practitioners with OSS experience. Since the OSS legal community is relatively small, compared to the OSS developer community [72], we adopted different strategies to disseminate our survey, trying to reach as many people as possible.

First, we posted the survey in Cyberprof, a listserv dedicated to Internet law educators. Second, we compiled a list of top legal firms in the United States, ranked by gross revenue, from the Am Law 100 list [21]. Potential participants were identified by manually visiting the firm websites and employee directories and collecting public contact information of people with relevant expertise, resulting in 724 email addresses. Third, we reached out to colleagues from our professional (SE and legal) networks. Fourth, we applied snowball sampling by asking potential participants to disseminate the survey to their networks, including private mailing lists. During interviews (described in Section 3.3), we also asked participants for people with whom we should share the survey.

3.2 Survey response collection and analysis

Responses from survey participants were collected using Qualtrics [24]; the survey was kept open for six weeks. Table 3.1 shows that 32 complete survey responses were obtained. Of these, 30 were valid; we discarded two invalid responses, one which skipped through the survey with single-letter responses and one which indicated a lack of experience in each answer ('I have little to no insight into...').

Twenty respondents self-identified as active in-house/outside legal counsel and ten as other roles. All have worked professionally in the area of OSS licensing, ten with more than 20 years of experience. When asked, "In your estimation, what percentage of your work involves software licensing?" twelve respondents indicated that more than 50% of their work involved such licensing. Figure 3.2 shows the distribution of all responses to this question.

Most participants were trained as lawyers (although not all were actively practicing); some respondents were not trained as lawyers but were actively engaged in OSS compliance efforts. All respondents came from our professional network, mailing lists, and snowball sampling. Notably, no lawyers from the top firms completed our survey, which illustrates the difficulties of surveying the OSS legal population.

We qualitatively analyzed the survey responses by applying a coding approach, in line with [71]. Two SE researchers and a law student (hereon "annotators") performed *open coding*, independently assigning one or more codes to each response using a shared spreadsheet. Each annotator independently coded all 32 complete responses, adding new codes to the spreadsheet when appropriate. Once open coding was completed, the annotators convened to settle disagreements and consolidate the set of codes. We did not base our analysis on inter-rater agreements since multiple codes could be assigned to each response and no list of codes existed prior to the start of coding. Where there was a high level of dispute or the response was ambiguous, an additional law researcher offered input. Where appropriate, results were analyzed by leveraging descriptive statistics on the assigned codes.

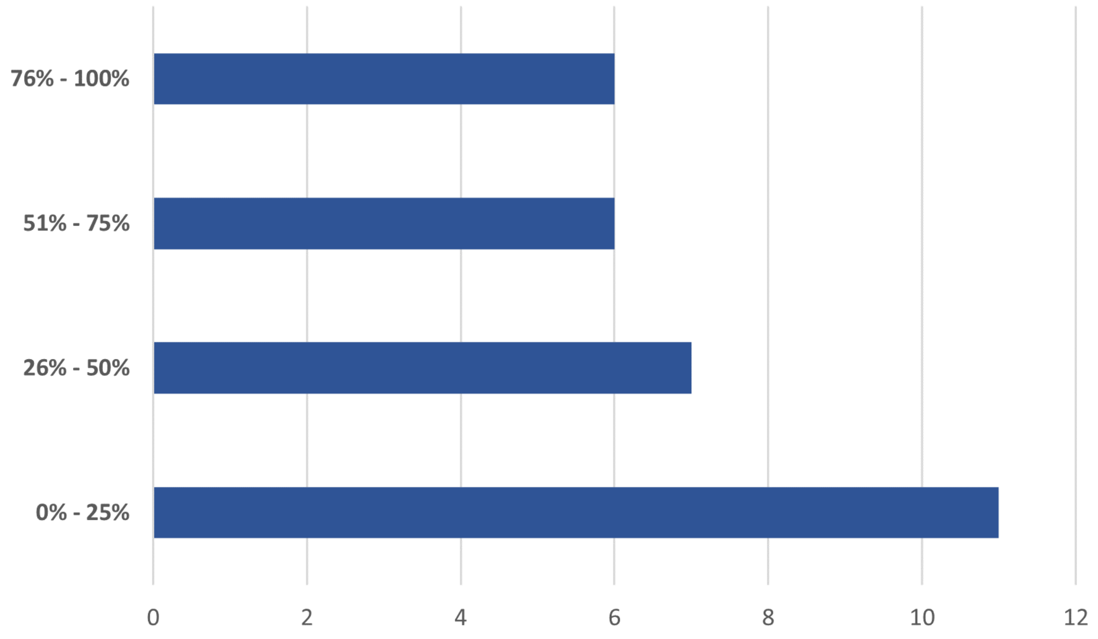

Figure 3.2: Percentage of practitioners' work involving licensing

3.3 Interview design and analysis

We conducted 16 semi-structured interviews over Zoom with survey participants who expressed a willingness for a follow-up conversation and whose survey responses warranted further investigation. Interviewees were selected to represent a range of backgrounds and professional experience. The interviews were designed to gather a deeper knowledge about respondent experiences and responses. Each interview lasted 30 to 60 minutes and was recorded to facilitate transcription and subsequent analysis. Both SE and law researchers prepared for and were present at each interview.

Each interview (recording and transcript) was independently reviewed by one SE researcher and one law researcher. A shared text document was used to aggregate findings from across interviews by assigning topic labels to all relevant portions of each response. To validate the accuracy and completeness of the final document, all reviewers, three in total, met to discuss and come to a consensus on the framing and application of topic labels.

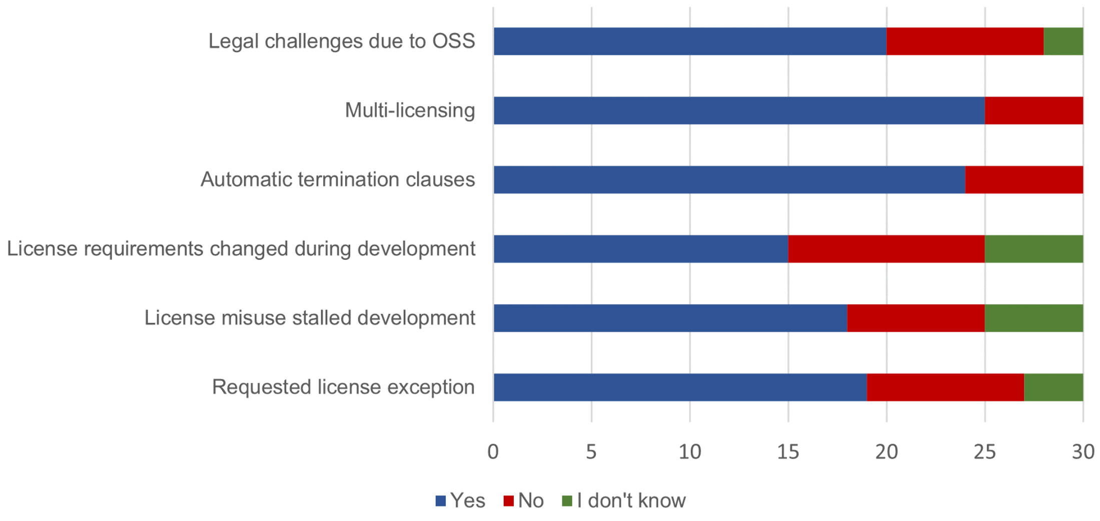

Figure 3.3: Practitioner experiences

Table 3.1: Responses and respondent demographics

Survey		**Interviews**	
Total responses	61	Contacted	18
Complete responses	32	Interviewed	16
Valid Responses by Role		**Interviews by Role**	
In-house counsel	12	In-house counsel	5
Outside counsel	8	Outside counsel	7
Other	10	Other	4
Total	**30**	**Total**	**16**

Other Roles	
Professor	3
Compliance Director / Expert	3
Executive Director	1
Enforcement Professional	1
Open Source Manager / Director	2
Total	**10**

Chapter 4

Results

4.1 RQ1: The Ecosystem of OSS License Compliance Experienced by Legal Practitioners

We discuss how OSS license compliance is generally experienced by legal practitioners. In particular, we discuss how licenses are selected and used, the nature of license violations, how licenses are enforced to address violations, and how disputes are resolved.

4.1.1 License Selection

Legal practitioners encourage OSS teams to use common, pre-drafted licenses (*e.g.*, those approved by OSI), as it reduces the effort required to understand a license's terms. One interviewee identified the benefits of being a "repeat player" and having a knowledge base built on the landscape of available OSS licenses: "One of the main advantages of OSS [licenses] is [that] they're templates.... These aren't negotiated every single time, so lawyering to some extent really comes down to the most basic skill of cutting and pasting." Without the need to draft a bespoke license, OSS developers seeking to license their software can focus on what they want users to be able to do with their software. However, there are a number of constraints in the license selection process. Notably, the licenses of the software's dependencies may limit the options a developer has to choose from when selecting a license, such as software licensed under GPL or other copyleft licenses which require that works based on them also use the same license [7]. One interviewee described their process for choosing OSS licenses for software: "[O]ne of the first things [we do is] verify that all the code was written by us. And then look at what the dependencies are of that code. Because it would be kind of stupid if we had dependencies on, [for example,] GPLv2 only to release ours under Apache. . . . And then ask the question, well, what do we want people to be able to do with it? . . . [D]o we mind if it gets used in proprietary software? . . . And then finally, what does the ecosystem look like that it's going into? Is it a particular community where one type of license is more popular than another?"

Finding 1: Legal practitioners encourage the use of off-the-shelf OSS licenses to avoid needing to analyze an individual license for each of a software project's dependencies.

4.1.2 License Proliferation

There may not, however, be an existing OSS license that exactly fits a licensor's needs. This could prompt OSS teams to create new, bespoke licenses, thus creating license proliferation [16]. Participants characterized this as undesirable as it increases license compliance efforts: "[W]hen someone takes one of the licenses and slightly modifies it, or tries to write their own license, then even when the license seems fairly innocuous, because it's an unknown, it requires more vetting by the powers that be." Interviewees seemed to agree, consistent with the views of OSS organizations [4, 11, 6], that writing new licenses is usually not necessary and even harmful: "I think the majority of folks have tended to say, 'We want to have as few new licenses as possible. The more licenses there are, the more compatibility issues there are.'" Additionally, drafting high-quality OSS licenses can be difficult, since doing so requires specific domain expertise: "[I]f you write [a license] without [an] understanding of how [OSS] development works, your license might not work . . . ." One way to address this challenge might be to simply modify existing licenses to suit one's own needs, but even this can obviate the benefits of off-the-shelf licenses and create documents that don't adhere to open-source values: "[W]hen they... try to white label the license... [and] get rid of the language they don't like... [then] it's not really that license that was meant to be used in its original form, it's something brand new. And... something that sometimes is not even open source." However, the consensus appears to be that license proliferation has waned in recent years; as one respondent noted, "[T]hat was a significant problem in the mid-2000s, but I don't see it happening anymore.... [E]verybody understands that one of the biggest benefits of open source is a known license that you understand the characteristics of and you don't have to read [to] figure out what it says."

Finding 2: Legal practitioners encourage the use of off-the-shelf OSS licenses. License proliferation can obviate the main benefits of off-the-shelf OSS licenses but is reportedly no longer a common practice.

4.1.3 License Exceptions

The off-the-shelf nature of OSS licenses also has implications for licensees, who might desire to engage in uses not covered by the applicable license. In these cases, legal practitioners can help their clients obtain license exceptions (or additional permissions), which allow clients to operate under different terms from the original ones provided in the license. This practice is not uncommon: 19 survey respondents indicated that they had experience negotiating or requesting license exceptions (see Figure 3.3). Experiences regarding such requests varied. Six respondents noted that exceptions were difficult to obtain, while five cited them as easy to acquire.

Respondents noted two challenge categories that may arise while seeking or managing license exceptions. One is the difficulty of contacting all relevant developers for a component. A developer may have died, or permission may need to be secured from multiple developers, depending on the number of dependencies. As one interviewee put it, "If you look at GitHub repositories, you would be amazed at how many stale links are on there, or how many people have moved or transitioned. . . ." The other is the informal nature of documenting exceptions, which one interviewee described as either an internal note or a statement in the repository of a public project: "[T]he technical person goes and contacts the author of project X and says, . . . ‘Could you send us an email just confirming you’re okay with x, y, and z?’. . . And then the client goes forward and just relies on that." Informal documentation can lead to difficulties when performing license compatibility checks or addressing potential license violations.

Finding 3: Obtaining license exceptions is not uncommon but can be challenging. It can be difficult or impossible to even start the process if project authors cannot be identified or reached.

4.1.4 Licensing Changes

Project maintainers may decide to change the license of their products (*e.g.*, from OSS to proprietary/business license [22] or between two OSS licenses [17]), for a variety of reasons, such as changes in business strategy [36, 76]. However, while not impossible, such license changes can be quite difficult for open-source software: "[Large scale license changes are] really rare because it’s incredibly difficult work because you need consent from all the parties who have ever contributed to the software." When license changes do happen, they can impact a project’s dependents. In some cases, the new license may not align with a dependent organization’s goals (*e.g.*, a closed-source project using software that requires source code disclosure). This leaves developers with two options: "[E]ither we switch to the new license and have to figure out whether we can [meet the new terms], or we stay on the old license [and forgo] security updates from the mainstream branch. . . ." A third option involves waiting for another party in the community to fork the original project and maintain it [61, 18], such as when "[p]eople forked [MySQL], and now there’s a competing product called MariaDB, which is GPL licensed."

Finding 4: License changes can happen for a variety of reasons, but they can be difficult to negotiate and can impact dependent projects. Proprietary relicensing can result in project forking.

4.1.5 Multi-Licensing

OSS teams can also decide to multi-license their products. Interviewees discussed the practice of multi-licensing, which was used to refer to various situations in practice, including software released under more than one license ("[M]ulti-licenses are generally easier to use because you have more than one license to choose from. And often it's a choice between a copyleft license and something that's usually more permissive."); software containing dependencies/files under different licenses ("there may be many dependencies, each of which is covered by different licenses"); and the offering of a free restrictive license with a paid option for a more permissive license (*e.g.*,, "You release under AGPL, and then some people can't use code under AGPL, so they want to buy exceptions from you.").

Multi-licensing can generate additional confusion when it is not clear whether multiple licenses apply at once or whether a choice is given between multiple licenses. This is sometimes referred to as determining whether the multi-licensing uses an AND or OR scheme, respectively. One interviewee noted how engineers attempting to interface with projects multi-licensed in this way "may read it as just yet another license they have to comply with versus it's a choice for this license and also many licenses. So it just creates an extra layer of complexity.... [H]aving to articulate the AND or OR statement just makes it that much harder."

The participants also expressed that a software project can be multi-licensed unintentionally if it is distributed through multiple channels under different licenses. One respondent described such a situation in which "On site A, it says that this is licensed pursuant to the LGPL [but on] site B, it says it's licensed pursuant to the MIT license...." It is unclear how often this happens.

Some participants also identified a concerning multi-licensing practice in which an organization may claim that a user of their software was out of compliance with its freely-available license in order to ask them to pay for a proprietary license, exemption, or the like. This is problematic in the case that the user in fact remained in compliance with the freely-available license: "The company would try to find people who were using [their software]. And then they would tell them, ‘oh, hey, I noticed you were using [the software] out of compliance with the license. Here’s what you can pay us in order to buy a license, so you don’t have to worry about that...’ It’s similar to patent shakedowns in the sense that they may not have a claim necessarily, but they ask for money anyway, just to see if they can get it with that threat."

Finding 5: The term *multi-licensing* has different meanings depending on context. When multiple licenses are available, tracking which license a component was received under becomes a challenge.

4.1.6 License Violations

The goal of license compliance is, of course, to avoid violations. In the 30 survey responses, several key themes emerged pertaining specifically to license violations, including conflicting license terms (6), missing source code required by certain licenses to be distributed (4), and software that is missing license information (3) or copyright information (1). We further explored such violations in follow-up interviews. Three interviewees indicated that issues tended to arise from software developers copying or using non-compliant code without considering its licensing information: "[P]eople find stuff on GitHub and they don’t pay attention to the fact that there’s no license in the repo and no license information in any of the source files. They just figure since it’s up on GitHub, it’s intended to be shared. . . . It happens weekly."

Multiple respondents indicated that they perceived license violations to be very common: "There is rampant non-compliance in the field," one indicated; another said, "[In] open source licensing, a lot of violations are ignored. People just kind of don’t care."

Some respondents characterized noncompliance as knowing and intentional, with compliance pursued only in response to complaints. Others indicated that being a "good citizen" was particularly important in open source communities and suggested that it was very difficult to prevent every violation. For example, when asked how organizations they have worked with ensure license compliance, one survey respondent indicated that "'[e]nsured' is impossible." Instead, compliance becomes a risk management activity in which risk might never be fully eliminated but can be mitigated, as one interviewee indicated: "[Tools] together with people who are knowledgeable on this stuff and processes that are designed to resolve problems can help organizations ... put together a process that is reasonably risk mitigating...."

Participants indicated that license violations can have important negative impacts not only on licensors and consumers (who might not receive source code to which they are entitled) but also on developers and their companies. As shown in Figure 3.3, 18 survey respondents indicated that organizations/clients they worked with had their development or release process stalled due to improper usage of licensed software. 8 of these indicated that licensing issues required software re-engineering and 6 indicated that such issues led to delays in shipping software.

Finding 6: Non-compliance is frequent in the OSS ecosystem, resulting, in part, from difficulties in maintaining compliance. License compliance violations impact non-compliant parties, software users, and the ecosystem at large. The development/release process can ultimately be stalled due to licensing issues.

4.1.7 License Enforcement and Dispute Resolution

When license enforcement occurs, our results show that it is typically performed by open source community organizations rather than directly by the parties whose licenses have been violated: "[C]ommunity compliance actions [are] by far the most common.... [Organizations like the Software Freedom Conservancy] are kind of like pro bono organizations." Resources for enforcement actions are limited, however, leading to differing enforcement levels based on the perceived importance of a violation: "The community focuses their limited resources on where they can make the biggest impact, which means that there's different treatment depending on who you are." Such enforcement actions can utilize one of several levers of enforcement. While many open-source licenses include automatic termination clauses, which terminate the license on breach, in practice, it is difficult to use these to achieve results quickly. One respondent explained that sending a cease and desist letter claiming automatic termination did not alleviate "difficulty in getting a preliminary injunction in federal court in the U.S... [since the] harm is primarily [to] the rights of users [and not someone] losing money."

License compliance disputes can be resolved without litigation. For example, enforcement organizations might extend the "cure period" in a license, the grace period where a non-compliant licensee can come back into compliance without risking license termination. (One interviewee cited a case in which a license had a cure period of 30 days, but legal action was not brought until two years later.) When asked how the organizations they have worked with resolved legal challenges related to the usage of open-source software, while litigation (7) and settlements (6) were mentioned frequently, a number of respondents also indicated that they resolved the matter informally with the other party (4), or by rewriting/removing non-compliant code (3) or otherwise bringing their software into compliance (2).

Respondents indicated that litigation is rare, and several interviewees identified that it is often viewed unfavorably in the OSS community. One interviewee noted that "there's a great deal of antagonism in the OSS community over whether or not lawsuits are appropriate at all. . . ," while another pointed out that "[l]itigation is an expensive process, and often results in decisions that may not achieve the [desired] result on either side." Thus, if a license issue is brought to a developer's attention, one option is to simply apologize and correct the issue, either by coming into compliance or rewriting the code: "I think normally falling on your sword and saying, look, we made a mistake here, and then working out some sort of solution is usually what happens." Such an approach seems to be effective in many cases: one interviewee noted that, in their experience regarding such notifications of potential license violations, they "would say the vast majority of [responses were] ‘thanks for pointing it out, we’ll fix it as soon as possible.’" Additionally, public shaming – issuing statements accusing an organization of flouting license compliance – can pressure organizations into complying to avoid reputational damage. As one interviewee put it, "Public shaming is a very effective strategy. . . . Technology companies. . . are in hot competition for engineering resources all the time. . . . And if you get known as an open source scofflaw, you have trouble recruiting people."

Finding 7: Enforcement is typically done by the community rather than by licensors. Due to limited resources, enforcement is unevenly carried out, with a perceived overall lack of enforcement. Litigation is rare, with other enforcement methods (*e.g.*, public shaming or informal resolution) used more frequently.

4.2 RQ2: The Process of OSS License Compliance from a Legal Perspective

We discuss different facets of the OSS license compliance process experienced by legal practitioners.

4.2.1 When Compliance is Done

Respondents indicated that compliance tasks can be done reactively or proactively. According to one interview, companies "approach license compliance as something that you do at the end to make sure that you’re able to combine all of these things in a way that meets all of the licenses." However, when this approach is taken for "whatever jumble of code you happen to have [at] the end, there’s a good chance there will be problems and they will be difficult to deal with." As seen in Figure 3.3, 18 of the 30 surveyed practitioners had experienced a compliance issue stall or delay a client’s product. Legal teams are acutely aware of the problem; according to one respondent, "[S]oftware folks are not real happy when [a component needs to be replaced] because you can't just immediately rip stuff out, rewrite it and test it when you've got a ship date that's very coming up very soon." The impact is heightened when copyleft licenses are involved: "[If you wait until the] end... and you find out you've got something that's GPL related..., you've got obligations to release the rest of the code along with the GPL stuff under those licenses. And you either have to do that or you have to pull it and find a replacement for the code that you have."

Alternatively, compliance can be done before or during the development cycle. This approach comes with distinct advantages. One interviewee mentioned that if you "do some assessments as you're going along in building the project, then I think it is much easier to both make the determination and then to comply at that point." Incremental scans and thoughtful compliance processes "help steer off those issues early on so that the team can say, 'Okay, this will be an issue if we use [that] dependency. Let's find a different one that's under a different license that's going to work better.'" Nevertheless, best practices typically still involve a final scan because "when you have a lot of hands in the pot, you don't know what people are bringing to the table and some people... might just slip something in there without having checked it in."

Finding 8: Compliance is less difficult and more effective if it occurs throughout the development process rather than solely at the end.

4.2.2 Roles in Compliance

Multiple actors/roles are involved in the compliance process, notably developers and lawyers. But who within an organization is responsible for carrying out compliance tasks? According to some participants, while the impacts of non-compliance are felt organization-wide, the ultimate responsibility is on the developers, since they are the closest to the code base. The role of legal teams is to "provide the tools that can help make compliance easier, the training that can help them understand why it is important, best practices or playbooks. . . . But ultimately it's the developer's responsibility. They're the ones closest to the code. They're the ones touching it." Another interviewee added, "I'm not searching through the code base as a lawyer on a daily basis and I, to some extent, have to rely on the developers to tell [me what's in the project]."

Finding 9: While developers and lawyers are both involved in compliance, as the ones closest to the code itself, developers have the ultimate responsibility of monitoring what components enter the software and maintaining compliance.

4.2.3 OSPOs

To alleviate the burden on developers and assist them in compliance tasks, some mid- to large-size organizations have Open Source Program Offices (OSPOs). These companies created them "because they started to realize they needed to coordinate the efforts that they were doing internally around open source, and they needed to understand how the appropriate ways to interact with open source communities were. The mindset of proprietary software didn't work when you were working within these communities." Most OSPOs include an "attorney that is a liaison with or part of that organization, who kind of gets the job of understanding the basics of open source licensing."

One interviewee described how their company's OSPO operates: "We've got some tooling that's used to scan software that we've developed before it goes out. And we educate developers on how they should go about doing what we call IP planning for their projects, which is basically understanding what the proposed distribution model is going to be. You kind of have to understand that at the outset to figure out what the guardrails are going to be. And then identifying any third-party software that they're planning to be using, making sure that it's under licenses that are going to allow us to use it in the way that we want to be using it. And then just sort of incremental checkpoints along the way." In addition to these roles, an OSPO can help ensure that a company has "the pulse of the open source movement," so that it is "doing things in a way that [is] productive and [does]n't cause backlash within [the] community that [it's] trying to work with."

Finding 10: Compliance is a team effort between developers, lawyers, and other roles. However, compliance is seen as a developer's responsibility. OSPOs are a way that larger organizations can manage license compliance and assist developers in the process.

4.2.4 Provenance and Recordkeeping

One of the major challenges in open source — something "software attorneys obsess about" — is determining what is in a project, where it originated, and what licenses attach to each component. Twenty-four of 30 survey participants indicated that provenance was essential information needed for compliance. One interviewee stated that if this information is not tracked at the time the component is copied, it can be "very difficult to go back and forensically reconstruct how they got [to] where they are now." Another noted that issues arise when developers "know that something is open source, but they won't know the nature of the license under which they grabbed it because [of old web links].... [W]e can't get the original one unless we go back in the Wayback Machine, which is hit or miss." Another respondent, a non-lawyer involved in compliance, had a contrary view: "I can tell you what's in your products within minutes of getting a report [of] a violation. It's not like folks have spent... time hiding it, and it's not like their upstreams are covering up the fact that there's Linux in there or any other copyleft program." (The different views may result from whether the task is to track all software components in a build or to locate a specific license violation associated with a single component.)

One way that developers can take on this responsibility is by keeping detailed and accurate records of where code and dependencies are pulled from. The need for recordkeeping becomes apparent in the situation where "it's years after the fact, contractors have left, [and] there's been a brain drain from the organization." The size of an organization has an impact on the importance of record keeping. "If it's a small shop, ... like four or five people, and they've developed all the software themselves and they know it inside and out, they rely on their own records. [Compliance is likely to consist of asking if] anybody remember[s] whether this has any third party software in it." Larger companies engaged in government contract work, on the other hand, need to keep detailed records because they want to avoid delivering a product "that [the government] can't use because it's got restricted software in it, or they wind up with, even worse, something that violates border control law where they're using software from overseas that [is prohibited]."

Even if the code's origin is clear, the nature of that origin can cause additional provenance problems, such as with snippets copy-pasted from other projects or pulled from coding sites like StackOverflow. Prior work has shown that snippets from such forums can be riddled with security vulnerabilities/bugs [25, 38, 69, 73] and lack attribution when reused [30, 28]. An interviewee commented, "[S]ometimes developers are doing exactly what we ask them to, which is to document where they're taking code from, [but] they ignore the fact that we say, 'Please don't take code from Stack Overflow.'" One interviewee offered a pragmatic perspective, stating, "If you pull a piece of code from some obscure... Reddit thread... you don't know if [that user] legitimately acquired that code."

Finding 11: Tracking software provenance is important for compliance tasks, yet is still difficult in some situations despite the available tooling.

4.2.5 Risk Management

Another important part of compliance is measuring and managing acceptable risk given the nature of open source enforcement. An interviewee who favors greater enforcement gave their take on the situation: "[N]o one who's in business to make money is willing to comply with the license without a penalty, and they do a calculus. And they say, well, okay, 'What are the odds that we actually get caught?' And the odds are admittedly low." Another interviewee told us in a hypothetical situation that "the truth is, because it comes out as a compiled product that is within an environment, the likelihood of the person that my theoretical client would have ripped off ever discovering that is slim to none." This view was not shared by all participants, with many stating that it was "very important" to their organizations "to be good open source citizens." As one interviewee put it, "[W]ith every launch there was inevitably some open source software developer that would reach out and say, ‘You’re not complying with my license,’ and then we’d fix it. I’m not sure most open source software developers are thinking of companies as targets. They’re more concerned about ‘Are you a good citizen?’ and ‘Are you trying?’"

A company’s risk tolerance may depend on where it operates. We were told that "the ‘move fast and break’ things concept is one that American technology companies are very comfortable with. . . . [I]t’s better to act and seek forgiveness than it is to seek permission. . . . We’ll just operate on this expectation. And if we need to buy somebody off on the back end, we will. . . . But in Europe, especially, and then in Japan, secondarily, they are much more risk averse when it comes to these things. . . . When I have a conversation [with] my French clients, as opposed to my American clients, as opposed to my Japanese clients, I have to . . . adjust a mental dial before I get on the phone with them where I say, ‘Hey, this is what I think is an acceptable approach for this customer, because there’s a different risk tolerance across the board.’"

As such, risk management strategies will vary between organizations. When survey participants were asked how organizations they worked with have managed this risk, responses indicated that automated tooling (9), training (6), and relying on counsel (6) were the most common strategies.

Finding 12: Risk management is essential during license compliance but may depend on organizational culture and geographical location/jurisdiction.

4.2.6 Education and Training

While there was an overall respect for software developers, many respondents noted that developers at times operate under false assumptions or incorrect information. One interviewee described "a very senior. . . developer telling us how various licenses worked, and [there were] some kernels of truth in what he was saying, but some of it didn’t actually line up with not just industry standards but what the licenses said." Another suggested that the experience of developers gives them greater confidence in their understanding: "Sometimes engineers, because of their experience or perhaps lack of experience in some cases, view themselves as quasi-lawyers... and probably rightfully so. In many cases, they have a lot more experience in open source software than a lot of lawyers... and might think, 'I know how this works.' Often they don't." Ultimately this state of affairs made one practitioner wonder what developers learn in school about best practices: "Are they taught that you're just supposed to go out and grab whatever from the web and move fast and break things?"

Executives, particularly C-level employees (*e.g.*, CEO, COO, CFO), make up another critical constituent group within a company which may require training regarding licensing. One interviewee noticed a disconnect between C-level employees (*e.g.*, CEO, COO, CFO) and development: "many times the C-level doesn't really know what their product does, or is capable of doing. I'm negotiating against sort of a larger customer... and say to [the executives], 'Look, the customer is asking for x, y, and z. Does it or can it do this?' And oftentimes, I get a shoulder shrug, and they have to go to the people who are actually coding." Another described that their "conversations with C-suite are very different [than those with developers]. They're strategic rather than tactical." As such, training with these groups may encompass strategies that involve utilizing both proprietary and open source software in ways that will "demonstrate overall value of the company to current and potential future shareholders."

Given the lack of background in OSS licensing, training developers is critical for any (moderately sized) company, as one survey respondent put it: "The number one form of risk management is developer education." Training can "arm them to make good decisions early in the process. So by the time [they get to the compliance review], they [have] a better chance of not having issues." Another characterized the message as, "You can't go out and create these [licensing] problems on the back end by saying, 'Oh, yeah, I just downloaded this. And it seems to solve the problem. Therefore, I'm going to push it out to our customers.'" At a high level, the objective of training is to "convince the engineers that the compliance activities are going to be reasonable and possible to do. And... to convince the lawyers that the compliance activities are going to substantially reduce the risk." Training is not meant, however, to push the responsibility of compliance solely on developers. "Training was really focused on just providing background information to engineers... so that they understood why we had the overall open source review and third party software review process.... It wasn't designed to have the engineers make the decisions on their own."

The success of training efforts varies, with some developers internalizing content and others seeing it as a requirement to be tolerated. One interviewee described training as a constant process, like "a game of whack-a-mole, where you're never catching up. I often describe my job as being kind of like in the movie Groundhog Day, where I keep having the same conversations with different people." Conversely, another interviewee believed that training "was effective in the sense that it brought general awareness and it got on people's radar that this was an issue and something they should be thoughtful about," but they acknowledged that it didn't give "each person... a detailed knowledge of the different licenses and [their] legal requirements." According to respondents, training sessions are typically recurring to ensure that the information stays fresh and are revised to reflect new developments, adapt to different learning preferences, or take account of areas where developers repeatedly have questions. In some companies, "enhanced training" exists for individuals who repeatedly or intentionally violate another party's IP.

Training is not the only way for legal practitioners to support development teams. Many also draft handbooks, licensing bibles, approval lists, or other forms of license clearance to offer engineers further guidance in their decision making [70]. (After training (19) and improved policies (9), licensing bibles (4) were the third most frequent educational method mentioned in the survey.) This guidance includes "extensive wikis, FAQ, [and] internal documentation about licenses," as well as "best practice documentations for our software team on things they should document as they're looking at open source technology." Although guidelines for which licenses are permitted and under what circumstances are sometimes referred to as an approved/denied list, a red/yellow/green light system, or a license approval matrix, these methods are more nuanced than their names imply. As one interviewee stated, "[I]t wasn't as simple as approved or denied licenses, but it was a spectrum, and we had pre-canned questions and advice based on what . . . parameters the engineers gave us."

All of these methods are aimed at helping developers and teams make better decisions before getting legal counsel involved: "I do have some green light, yellow light, red light stuff, clearly delineated on those lists, so [the developers] understand . . . when they need to pick their head up and have . . . a proactive conversation with me, before they start to incorporate technology." In other words, these resources support a "self-service model [of] education, [where legal counsel doesn't] have to be integrated in the entire process, and [the engineers] can educate their colleagues. . . ."

Finding 13: Developers may have an incomplete or incorrect understanding of compliance requirements, making training essential. Since it can be difficult to get developers invested in training, lawyers can create additional resources (*e.g.*, guides and bibles) that help developers with compliance tasks.

4.2.7 Tooling

Tool usage is often employed in compliance tasks, particularly to detect and track third-party components. Of the 30 respondents, 21 indicated that they (and developers) used tooling. The most popular offerings were Black Duck [2] (11), FOSSology [3] (4), and ScanCode [12] (3). As one interviewee said, "These days, you really can't do [compliance] without tooling. . . . [M]ost sophisticated products have way too many components for anyone to keep track of them by hand."

Respondents reported that tools are primarily used during the development process or prior to shipping software to validate license compliance. During acquisitions, code scans are used to "determine whether any of that open source code is subject to one of those copyleft licenses" as part of a risk mitigation strategy. Organizations and developers can also use "binary scanners [to] identify [their] open source code in [a proprietary software product] [using] that as a mechanism to tell [the distributor] that they're not complying with license obligations."

Through our interviews, we distill three main steps in the compliance process: 1) identifying third-party components being included, 2) determining the licenses of those components and how those components will be used, and 3) applying the client's compliance policies to that information. One interviewee assessed that tooling is "especially essential for step one, partly essential for step two, and not really that relevant to step three." Similarly, another respondent stated, "I trust the computer's ability to match code better than I trust the human."

A few respondents described the current state of tooling as "pretty good." One respondent was "not aware of too many situations where clients have used a tool like that and it missed something that was open source." Another said clients use tools because they "are pretty good in terms of having massive databases of open source code against which they compare, [which gives] a high degree of confidence that you've identified third party open source code that's part of your stack."

Nevertheless, tools are still imperfect. One of the primary limitations mentioned in our study was the accuracy of the tools. One respondent said, "Probably more often than not, [tools are] accurate. But when [they're] inaccurate, it's really painful." Another respondent noted that in their experience tools "usually don't capture [meta information] properly," which is why they "never rely on the tools for that information [because] most of the time it's inaccurate."

One of the main inaccuracies we encountered in our discussions was false positives, which can cost a team or organization valuable time. Respondents discussed cases where tools were "reporting as a match things that wouldn't be eligible for copyright protection anyway, . . . like class names." Tool output inaccuracies may result from how the tools are configured. As one respondent put it, some tools have a setting "where you can search for key terms that you find problematic, and that can be really noisy depending on how you configure it." One interviewee reasoned that tools are "just being overly cautious," which leads them to "identify a lot of false positives, things that they're saying [are] in the code base and then when you actually dig into it with your software engineers those pieces are not in the code base." Another type of false positive presents itself as a "license problem when really it's just a missing word or misidentified license." Determining if a detection is actually a problem "requires a much deeper dive." These false positives can result in financial loss since organizations "have to pull these very busy developers off all the things they're doing and say, 'Hey, let's go through this list.' It's the last thing on earth they want to do."

Another reported limitation of tooling is the ability to provide analysis of compliance issues. Tools are generally good at providing information, but human review is required to know what to do with that information. One respondent described it as "quite frustrating because the output of the tools is information but not answers," and another said the current capabilities are "a far cry from human knowledge." Additionally, "most of the tools don't have awareness of how the code is integrated and used within your proprietary code, [making it] really hard for the tools to make decisions about whether something actually complies with the license." This leads to a "tendency for automated tools to [do a] red, yellow, green categorizing," which "misses a lot of things and can lead clients in the wrong direction on either end of the spectrum." Some respondents even reported that tools seem "to ignore important parts of the license and focus on parts that are still important but not the parts we see most commonly violated."

As described in Section 4.2.2, there was a general consensus among respondents that developers should be the primary tool users. One respondent said, "The tools absolutely need to be for developers. . . . When you're choosing software, it's not just a licensing decision. It's a security decision, code quality decision, and maintenance decision. And so it's primarily really an engineering decision. Licensing is just one of those pieces." Nevertheless, respondents indicated that any user of a tool needs to be aware of its limitations. One interviewee noted the importance of tool developers being "clear with their users . . . [who need to] have the understanding that none of these are the silver bullet [and that] these are all solving different pieces of a much bigger question." Tools can also be difficult to setup and use. “[Some] are not tools that you can simply deploy and they run themselves. . . . [T]hey require dedicated teams to manage the outputs and triage the results.”

Interviewees suggested several improvements to tooling. One stated, "Better maintenance of the upstream location where the tool is pulling information from. A lot of times there'll be dead homepages. It's not kept up to date. They'll put something into the catalog and then a couple of years go by, and it's moved location and it takes some human engineering to figure out where the code currently is or what the status currently is." Another would like to see tools that "produc[e] a really good notice file, not just like a phone book-sized SPDX dump, but actually something that's human readable and useful." Lastly, there was a call for tools "that can do both license and security checking at the same time. It would be beautiful if we only had to scan a code base one time with one tool." As one respondent commented, "The one thing that I think would be of great use is one tool to rule them all, and one tool to rule them all that doesn't cost an arm and a leg."

Finding 14: Organizations commonly use tools to identify software components and their licenses. However, tools provide little to no explanation and analysis of licensing issues. The number of false positives must be reduced for tools to be more effective and practical.

4.3 RQ3: Challenges that Legal Practitioners Face During License Compliance

Respondents described a number of challenges that legal practitioners face during license compliance. We focus here on changes in the way that software is developed and distributed; interpretation of license language; and communication between the legal team and the engineering team.

4.3.1 Changes in the Way Software is Developed and Distributed

The way software systems are integrated and built has changed over the years, with more and more components being integrated into new systems. The number of components and dependencies typically involved in even a small project has a direct effect on a lawyer's compliance work. Each component must be verified to ensure that its license is being complied with, and the decision is ultimately one that requires human review. One interview respondent recounted an experience in which the client went through a significant acquisition, which involved running a scan of all involved components; the resulting list of components alone was 50 pages long. The interview respondent concluded, "It is honestly impossible to do an effective review of that. We could evaluate and say, 'OK, fine. These things are important; these things aren't. But just as a practical matter, there is no useful tool that will determine whether or not something infringes." Another interview respondent conveyed that the licenses can create "an enormous administrative burden. Don't get me wrong — it was perfectly reasonable to do, particularly at the time open source licensing started, but I'm not sure that even the most ardent open source or free software advocate in the 1990s contemplated that people would be selling products that had thousands and thousands of components." This has the potential to pose a particular burden on smaller companies: "[M]ost companies care about being compliant, but, honestly, very small companies are just flying by the seat of their pants and their compliance is all just in time.... [T]hey just can't afford compliance activities or compliance tools."

Finding 15: Software development and systems have become more complex over time due to the increasing number of components/dependencies, making lawyers' compliance efforts harder.

4.3.2 Term Obsolescence

As described in Section 4.1.1, legal practitioners' proclivity for choosing from off-the-shelf OSS licenses has canonized several prominent licenses. However, in light of the changing software landscape described above, these licenses themselves age and some terms grow obsolete. One interviewee pointed to current notice requirements as an example: "the whole point of those notices is to inform people that the authors of this software licensed this software. You could just as easily for instance, if you wanted to know about the software that was in a product you had, and someone gave you a webpage and said here, you can find that information here, that would be perfectly useful for most people. That's not what the licenses require. They require you to deliver full-text copies of the license and so forth... [as] when most of the licenses were created, the web, it existed, but it wasn't available to most people." As the interview respondent above communicated, "Clients ask me all the time, 'Why do I have to [provide notice files with the software]? Does anyone look at these? Can't I just post them on a website? And the answer to that question should be yes, except for [the fact that] GPL v.2 was written in 1991." In this respondent's view, the situation is unlikely to change due to "divisions in the community" that cause inertia and paralysis.

Finding 16: Many currently-popular OSS licenses were first drafted years ago, leading to some debate over whether certain older license terms are still valuable. However, changes to these terms currently seem unlikely.

4.3.3 Interpretation of License Language

OSS licenses are documents intended to have legal effect, with the potential to be enforced in court. As with other legal documents, parties may have different interpretations of ambiguous terms in an OSS license. As one respondent noted, "[T]he law doesn't work like a computer does, [where] you put inputs in and you know what the output is."

This interpretive difficulty arises in part, some interview respondents suggested, because most OSS licenses were drafted and promulgated by software engineers and others in the SE field and not by lawyers. Thus, as one survey respondent noted, such licenses "sometimes are written using terms that don't directly map onto legal doctrine or are just ambiguous (perhaps intentionally), and questions then arise [as to] how best to understand them." This causes what one interview respondent stated to be "one of the downfalls of OSS" —

"the text is static in time, but obviously technology is moving," sometimes resulting in a situation where "the authors [didn't intend] the license to have this impact, but it does just because the wrong terms were used given the new technology."

As one manifestation of this situation, one interview respondent highlighted a multi-licensing scenario in which the copyright owner purported to license the software under two licenses simultaneously — that is, with an AND operator rather than an OR operator, as described in Section 4.1.5, which would give users the choice of using either license. But the challenge with having two licenses apply from a legal perspective is that "sometimes there's conflicting terms, and it just doesn't make sense. I don't know how you reconcile that, because there's just lack of clarity as far as what the license terms are." Notably, the SPDX specification includes expressions to represent such license combinations in an SBOM [13]. However, while this standardizes a representation of these scenarios, it does not resolve the associated intepretative questions.

Two examples of interpretive uncertainty were mentioned by interview respondents. First, the requirements of GPL v.2 apply to modifications that form a work "based on the Program," which GPL v.2 defines as "any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language." The question then arises whether, for example, an application under GPL v.2 that makes a system call to a library creates a unified work "based on the Program" or whether the library remains a distinct program. The GPL FAQs (Frequently Asked Questions) state [5], "Linking a GPL covered work statically or dynamically with other modules is making a combined work based on the GPL covered work. Thus, the terms and conditions of the GNU [GPL] cover the whole combination." But U.S. copyright law is not as clear on this question, as cases such as *Lewis Galoob Toys, Inc. v. Nintendo of America, Inc.* [15] illustrate.

The second example involves when an activity includes distribution, which is the trigger for compliance in several licenses. Interview respondents who raised this issue noted distinctions between products that are provided to end users versus those used only internally (which are arguably not distributed), although SaaS [35] complicates this question. One interview respondent gave another scenario: "[The GPL] says [that] if you distribute software that includes an GPL component, then your distribution has to be covered under GPL. But if you distribute something that works with something else [that] gets called at runtime but it's not part of your code, and there's a dynamic link or some other kind of communication between the programs, to me, you're not distributing that GPL program. . . . So there's issues like that [one] that are pretty significant."

The presence of ambiguous terms is not a fatal obstacle to enforcement; it is rare that a legal document is not open to interpretation in some respect. But respondents reported that because relatively few disputes in the OSS licensing space are resolved through litigation, there is a paucity of caselaw from U.S. courts on how OSS license terms should be interpreted, compounded by the existence of many unresolved issues in U.S. copyright law generally. This means that lawyers look to community resources and norms as a guide to license interpretation. These norms are particularly relevant because, as one interview respondent described it, "it's less about legal sometimes and more about just playing nice with the community. That dynamic is very, very important."

FAQs and similar publications developed by license stewards were reported by respondents to be a primary source of authority on OSS license interpretation. FAQs by the Free Software Foundation [5] were seen as particularly influential, even though, in one interview respondent's view, they were "long, complicated, and old." In addition to FAQs, interview respondents mentioned other sources of community beliefs of license meaning: listservs, mailing lists, and other community channels; sources such as Hacker News [9] (a social news website run by the startup incubator Y Combinator); and information posted on GitHub and other documentation sites by developers.

Although the community might look to the views of license stewards as to the meaning of the licenses they drafted, it is unclear whether a court would also do so. One interview respondent put it this way: "If you're just cutting and pasting a license that the Free Software Foundation wrote or Mozilla wrote, who's the drafter? Whose intent actually matters at that point? Because, sure, the Free Software Foundation is the author of that license, but they're not the licensor.... Are you [now] the drafter because you chose what to cut and paste? Does your intent as the licensor matter? That becomes a really interesting question."

As a practical matter, the lack of interpretive guidance from courts, and the shift in interpretative authority to the community, affects the nature of the advice lawyers give to their clients. In some cases, attorneys have to explain to clients that the community's view may not be cognizable in court; in other cases, compliance becomes a matter of conforming to community norms rather than conforming to legal requirements. One interview respondent stated, "[W]hen a client comes to me and says, 'What I should do?,' I advise them what to do in order to meet community practice. I can't really advise them what to do to meet the letter of the law, because the letter of the law is too vague."

Additionally, it is difficult to know in advance whether a particular norm will apply equally to all participants. As one survey respondent stated, "[M]aybe a competitor can do X, Y, and Z, [but] we might not be treated the same. So we would have to factor that into our analysis." Similarly, since there is no single arbiter of interpretive uncertainty, whether the "correct" answer has been reached in any particular case is left up in the air. One survey respondent characterized it as follows: "[C]ertainly I and my clients set out with the goals of conforming to OSS licenses. But even with our best efforts, we can't be sure that we are honoring either the intent or the letter of what the licensor is trying to do... [I]f you look at some of the open source communities and the sources for some of the licenses, they will offer their gloss as to what they think [the language] means. But... absent a court's interpretation of how that language would be honored by a court in the case of litigation, we're all just kind of guessing...."

As a result, lawyers might, as a form of risk management, advise clients to steer clear of any interpretive gray areas. One interview respondent stated, "My experience talking to other attorneys in the space has been that, for the most part, no one really wants to push the limits too much.... [T]o the extent the ambiguities exist, they kind of want to [avoid] them. . . ." Another interview respondent noted that although the client might have a strong legal argument, "maybe we don't want to be the ones defending that argument. . . . [It's] easier and simpler to just align with the community's view on the topic than to get into a discussion and try to walk in that gray area." On the other hand, this respondent acknowledged that there are times when the client needs to take a stand: "[It's] hearing people out, trying to educate, trying to understand, 'Is there a middle ground?' But some cases aren't just as simple as, 'Oh, we'll just fix it and move on.' Sometimes it is fundamental to the architecture. We just fundamentally disagree."

To be sure, not every interview respondent shared a similar view on the interpretive difficulties of OSS licenses. One interview respondent who is involved in OSS license compliance (but who is not trained as a lawyer) conveyed the view that the obligations under many OSS licenses are clear and that entities that claim interpretive uncertainty are "trying to figure out what the rest of the industry is doing" when they should instead "make a conservative interpretation" of ambiguous language. Another interview respondent stated that in transactional work, ambiguities "don't come up that often" because "there is a pretty good consensus among [OSS] attorneys, at least in the United States, about how these licenses work in the vast majority of cases." A third interview respondent who also favored strong enforcement of OSS licenses characterized interpretive debates as "a political struggle between firms who want certain licensing outcomes" and those who "care about the rights and freedoms of users."

Respondents generally seemed to be aware of this diversity of views, the result of the fact that the OSS legal community is relatively small. Indeed, one of the interview respondents mentioned above expressed concern about whether things would change as more commercial entities "who don't have an interest in the license" incorporate OSS into their offerings. Another interview respondent noted, similarly, "The community takes a different view on some topics than a large software company," highlighting, in this phrasing, that large software companies are, for some, not seen as part of "the community." Attorneys in this field can therefore gain their own interpretive authority, as one interview respondent stated, "merely by being around for a long time."

Finding 17: OSS licenses pose interpretive challenges because licenses are static while technology evolves over time, licenses were drafted by SE/CS experts rather than lawyers, and there is a lack of interpretive guidance from U.S. courts. As a result, lawyers rely on community norms and best practices (documented in sources such as FAQs and mailing lists) to interpret licenses and provide legal guidance to their clients. Lawyers sometimes advise clients to avoid "grey areas" due to interpretive uncertainty.

4.3.4 Communication Between the Legal and Engineering Teams

Effective OSS license compliance ultimately depends on productive relationships between lawyers and developers. Five interviewees mentioned a background in SE or a related field, which they believed facilitated communications with engineers because it gave them a certain level of credibility when discussing SE issues. One such respondent reported that they would use this background to write "toy programs" to illustrate to software engineers the licensing impact of certain development decisions because "if you can't talk to those engineers, if they don't understand you, [if] they don't have that respect, your advice really isn't going to go anywhere." By contrast, one interview respondent, who did not have a software engineering background, described using an intermediary to facilitate communications with software engineers employed by their clients.

A particular challenge reported by respondents was a need for developers to understand the value contributed by legal counsel. One survey respondent, who is not trained as a lawyer but consults in the compliance arena, stated that "the worst" situation "is when there is a local 'tech guru' who thinks [they know] how the licenses work but [who] has never actually read the actual license texts, and everyone in the company goes to that guru. It is super counterproductive." Similarly, lawyers at times may need to explain that what may be recognized as a harm by the community may not constitute any legal wrongdoing. One interview respondent gave an example: "Someone may get very incensed because another person added a copyright notice above them, making it look like that other person made a greater contribution than they did when they really had a greater contribution. [But] that is perfectly lawful under copyright law; it's even perfectly lawful under the license... [When you] explain to them that they actually don't have a legal claim for that... they're sort of surprised." A related challenge arises from cultural differences in attitudes toward intellectual property that originate in a developer's past experience. As one interview respondent stated, "Sometimes... getting [people] to appreciate the value of IP and the importance of respecting other people's IP can be an interesting opportunity or a challenge. They don't see anything wrong with [what they're doing], or they've been like, 'Well, I've been developing all this time doing this thing.' And it's like, '[W]ell, we don't do that here.'"

Ultimately, interview respondents reported a need to ensure that developers and lawyers know that they are on the same team, both in terms of compliance and in terms of the shared business interest in putting out a good product. As one interview respondent put it, "the key to being a good lawyer is becoming a... partner rather than a police officer. You are there to enable them to get their work done. Sometimes you have to say no just based on policy, but other times, you've got to try to use creativity to help them get to where they're trying to get." One survey respondent, who is not trained as a lawyer but who works on compliance issues, expressed skepticism about this relationship, stating, "In my experience, what I've seen is organizations tend to be told by their legal departments to 'just trust us' that we know how to comply with the license." An interview respondent offered a similar thought, noting, "I think for a lot of companies, once they get a lawyer involved, they tend to outsource a lot to the lawyer and just assume there's standard forms and clauses that everybody just accepts." A survey respondent, however, offered this perspective: "The legal teams establish policy and provid[e] guidance, while the engineering teams have to take that guidance and apply it to particular technical use cases. Without trust and open communication between those teams, it is impossible to resolve [licensing compliance] challenges."

Finding 18: Communication and trust between lawyers and developers can be challenging, in part, due to a lack of developer understanding of the value of legal advice, different cultural attitudes toward intellectual property, and different company norms about how lawyers and developers should interact.

Chapter 5

Threats to Validity

External Validity: The conclusions drawn in this study apply only to the population that participated in our survey and follow-up interviews. We cannot generalize our results, but in light of the themes that emerged, we believe that other lawyers in the OSS community will share some of the same experiences. That said, our goal was not to claim generalizability but to attain a fuller understanding of the software compliance landscape from a legal practitioner's perspective and to identify current practices and challenges. **Internal Validity:** To mitigate bias, we used an open coding methodology for both the survey and interview transcripts and had SE and legal researchers involved in every stage of the process. We employed diverse strategies to locate participants (professional networks, mailing lists, top law firms, *etc.*) to increase the pool of different perspectives and minimize potential bias, but we are aware of the issues that may arise from low response rates and self-selection bias. We followed best practices in the formulation of survey and interview questions, making sure that questions were written clearly and concisely to avoid confusion and avoiding biasing language. Interviews were limited to at most an hour, meaning that not all of a participant's views were likely heard. We limited confirmation bias in qualitative analysis by independent coding, discussing disagreement, and arriving at a consensus based on the data.

Chapter 6

Related Work

License Compliance, Practices, and Needs. Given the challenges and importance of license compliance, researchers have developed a number of tools and processes to assist with license compliance tasks (*e.g.*, detecting and fixing license incompatibilities) [39, 74, 65, 49, 42, 40, 41, 47, 37]. Other works catalog and detect non-approved licenses, license variants, exceptions, and questions by mining software repositories [60, 82, 66, 78]. Prior work also explored when, why, and how developers change their software's licenses [36, 76], as well as ways to predict license changes [55]. Surveys and analysis of Q&A websites reveal that developers tend to have difficulty understanding OSS licenses [27, 26]. Other work addresses this by investigating the automatic summarization of legal documents [58] and proposing systems of recommending licenses [48, 56]. All this prior work does not examine the state of the practice of license compliance from a legal perspective, as we do.

Licensing Bugs and Violations. Prior work cataloged licensing bugs and incompatibilities based on analysis of OSS project repositories [77, 80]. Several studies also indicate that licensing bugs are prevalent in modern software ecosystems, such as the NPM, RubyGems [57, 60], Android [62], PyPI [81], and JavaScript [68] ecosystems. Additional work also shows the prevalence of multi-licensing in JavaScript [63], further complicating the licensing landscape. All this prior work has focused on repository mining, while we conduct surveys/interviews with legal practitioners.

Chapter 7

Conclusion

By qualitatively analyzing the survey and interview responses of 30 lawyers and other individuals who specialize in OSS license compliance, we have identified: (1) the state of the ecosystem of OSS license compliance, (2) how legal practitioners perform compliance, and (3) the challenges faced by legal practitioners during compliance. Our findings warrant further research intended to support developers, legal practitioners, and other roles in performing license compliance more effectively, which is essential for software engineering companies and the OSS community. We now discuss the implications of our findings.

Robust tooling is needed. Compliance has become more difficult at the scale of modern software, where a single product might contain thousands of components. Given the enormous scale of compliance tasks today, manual compliance analysis is likely infeasible for large software products. As such, the importance of automated tools to assist with license compliance only increases. Yet, as our participants identified, current tooling can be difficult to use, is prone to false positives, and is limited to providing data rather than analysis. This demonstrates a clear need for more robust license compliance tools that are accessible to both developers and legal practitioners, keeping in mind the inherent analytical limitations of any tool.

Effective communication and interaction between lawyers and developers is needed. The optimal process is not for developers to build a software product and then pass it on to lawyers for compliance. Instead, lawyers should be involved throughout the process. This requires effective communication between the teams to ensure an understanding of each team's domain. Legal professionals can facilitate this by creating educational and actionable resources for developers so that they can be aware of the legal implications of development decisions and can more independently make decisions that will not negatively impact the organization at large. Indeed, our study suggests that similar educational efforts should take place early in developers' careers – during their undergraduate and graduate training – so that legal compliance is understood to be an integral part of software engineering, not separate from it.

License compliance should be integrated and continuous. We saw in our conversations that proactive approaches to compliance that are done early and often lead to the best outcomes. Put another way, from the inception of a project, *compliance should be treated as a nonfunctional requirement.*

Compliance tasks should start before the first lines of code are written and should be regularly assessed during the development process. For example, the compliance obligations of third-party components should be vetted *before* incorporating them into a build. Regular scans can also verify that no dependencies were slipped in under the radar. Tool-based license compliance checking can be incorporated into continuous integration/delivery pipelines to detect potential licensing issues in the process. Additionally, well-understood agile development practices can also be easily adapted to facilitate license compliance tasks. For example, if records are kept during sprint meetings, it can be much easier to track engineering decisions and software provenance.

It may take time for developers to adjust to a more compliance-minded approach, as some software developers, particularly those who work in small teams or on personal projects, have the mentality of "move fast and break things." Future work can explore the information needs and requirements for an *integrated and continuous compliance process* that works in tandem with development processes.

Community norms influence license interpretation. Software has changed significantly since many OSS licenses were first drafted. There appears to be little interest, however, in bespoke licenses with updated terms; legal practitioners strongly prefer using an existing license. In addition, a lack of case law facilitates the use of community norms as a source of interpretive guidance, but also means that ambiguities in licenses will persist. OSS license interpretation will therefore be a significant challenge for the foreseeable future, which increases the likelihood of (perceived) noncompliance. Going forward, the risk of interpretive gray areas can be addressed in several ways, including attention to known ambiguities by the drafters of future license revisions; continuing the trend away from license proliferation; and greater collaboration and convergence on published sources of interpretive guidance.

Chapter 8

Bibliographical Notes

The paper supporting the content of this thesis was written in collaboration with other members of the SEMERU and SEA research labs in the Computer Science Department at William & Mary and researchers from William & Mary Law School. It is currently under review for publication.

Wintersgill, N., Stalnaker, T., Heymann, L., Chaparro, O., & Poshyvanyk, D. (2023, September). "The Law Doesn't Work Like a Computer": Exploring Software Licensing Issues Faced by Legal Practitioners. Under Review.

Appendix A

Survey Questions

Below, I provide the questions asked in the survey in full. The questions are divided into categories: demographics (D), clients and client practices (C), experience (E), edge and special cases (EC), and tools and information needs (N). Additional details on the responses to each question can be found in the replication package [64].

(D1) What is your current role?

(D2) In your estimation, what percentage of your work involves software licensing?

(D3) Please briefly describe your experience in negotiating or monitoring compliance with software licenses.

(C1) How have the organizations that you have worked with ensured compliance with the licenses of third-party software (components and systems) used in their own software products?

(C2) How have the organizations that you have worked with managed the risk associated with using open-source software that is licensed under different licenses?

(C3) What types of software licensing issues have you encountered in your career?

(C4) Were some issues more difficult to resolve than others? Please explain.

(C5) Have the organizations that you have worked with ever faced legal challenges related to the usage of open-source software?

(C6) How were these challenges resolved?

(C7) How have the organizations that you have worked with educated their employees on software licensing and intellectual property rights to ensure compliance with licensing requirements?

(E1) Have you ever been involved in negotiating or requesting a software license exception?

(E2) Was the exception easily given/obtained or were the negotiations more complicated? Please, where possible, explain any complications.

(E3) Have any clients or organizations that you have worked with had their development/release process stalled due to improper usage of licensed software, images, fonts, and/or databases?

(E4) Please briefly explain how the improper usage led to such stalls.

(E5) Have you ever advised a client who was working on a project where the licensing requirements changed during the course of development?

(E6) What impact did the changing requirements have on the project?

(EC1) To what extent have software licensing issues arisen in your work related to operating in different jurisdictions, either in the United States or worldwide?

(EC2) Do you have experience with automatic license-termination clauses (the ability for an organization to revoke a license at any time)?

(EC3) Please describe your experience with automatic license-termination clauses.

(EC4) Do you have experience with multi-licensing software projects (software released under more than one license)?

(EC5) Please describe your experience with multi-licensing in software projects.

(N1) What information do you typically collect in order to make a determination of whether your client or organization can legally incorporate another entity's software into your client or organization's own product?

(N2) What difficulties (if any) have you faced in collecting this information?

(N3) Please describe the process you use to determine whether any software licenses issued by your client or organization have been violated.

(N4) What difficulties (if any) have you faced in connection with the process you described?
(N5) Are you aware of or have you used any automated tools that provide support or assistance in resolving software licensing issues?
(N6) What features would a tool need to have in order to help you with addressing or giving legal advice about license compliance issues?

Appendix B

Image Credits

This thesis contains images sourced from flaticon.com, used with permission.

Table B.1: Icon Attribution

Icon	Author	URL
Professional Network	Freepik	https://www.flaticon.com/authors/freepik
Top Law Firms	Freepik	https://www.flaticon.com/authors/freepik
Mailing Lists	surang	https://www.flaticon.com/authors/surang
Snowball Sampling	Freepik	https://www.flaticon.com/authors/freepik
Survey	Freepik	https://www.flaticon.com/authors/freepik
Response Coding	Freepik	https://www.flaticon.com/authors/freepik
Qualitative Analysis	monkik	https://www.flaticon.com/authors/monkik
Interviews	Freepik	https://www.flaticon.com/authors/freepik
RQ1: Ecosystem	Freepik	https://www.flaticon.com/authors/freepik
RQ2: Process	Gregor Cresnar	https://www.flaticon.com/authors/gregor-cresnar-premium
RQ3: Challenges	Mayor Icons	https://www.flaticon.com/authors/mayor-icons

Appendix C

Additional Survey Data

This appendix details additional response data from the survey's multiple choice questions.

Figure C.1 shows participants' responses to question N1, "What information do you typically collect in order to make a determination of whether your client or organization can legally incorporate another entity's software into your client or organization's own product?" Participants could select more than one response. Participants specified the following "other" responses: N/A (2); List of dependencies (2); Copyright attribution (1); Known bugs and security issues (1); State of Project (active, maintained, etc) (1); The Licensor (1); If OS will be modified (1); License URL (1); Presence of sub-content with different license (1); Patent / Copyright obligations (1); Whether compliance is even possible (1); Customer risk tolerance (1); Customer use case (1); Source code (1); License agreement (1); How software interacts with primary software (1); Dependency licenses (1).

Question N5 asked participants, "Are you aware of or have you used any automated tools that provide support or assistance in resolving software licensing issues?" 21 respondents answered "yes", while 9 answered "no." Participants who selected "yes" were asked to list the tools they were aware of or had used. Figure C.2 shows the tools practitioners identified.

Further details on participant responses, including the results of the open coding of open-ended responses, can be found in the replication package [64].

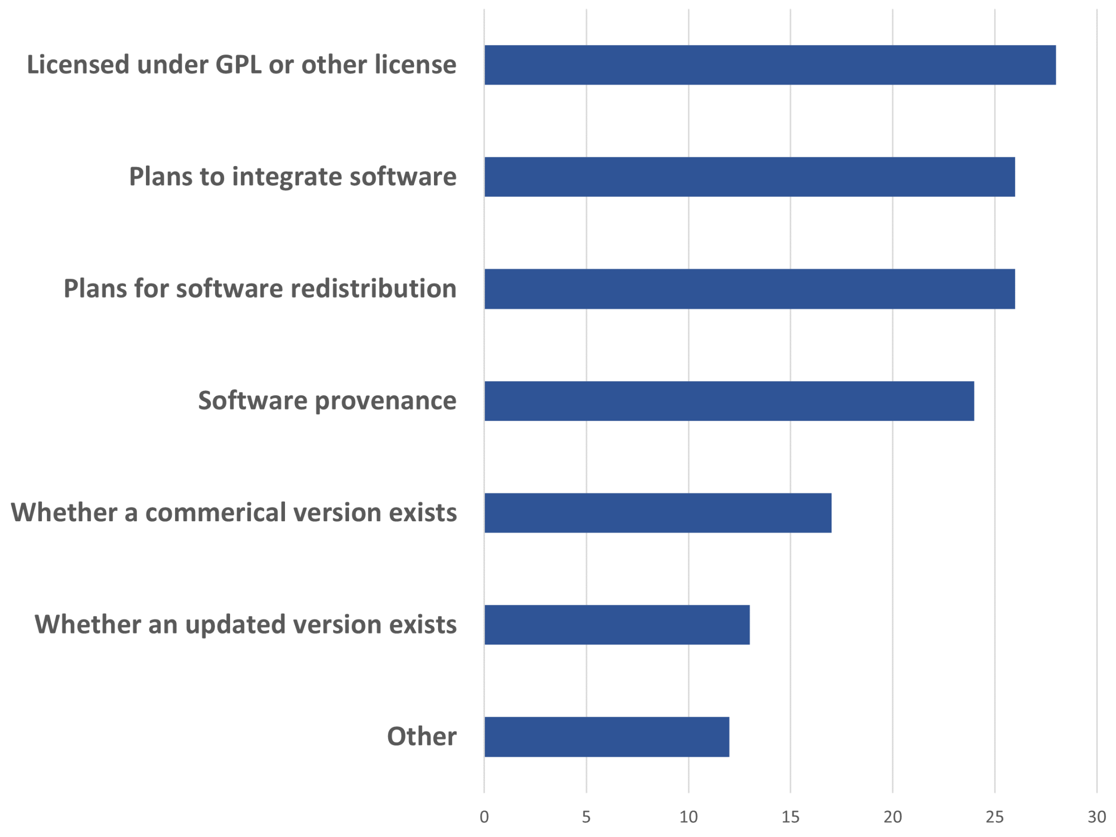

Figure C.1: Practitioners' information needs

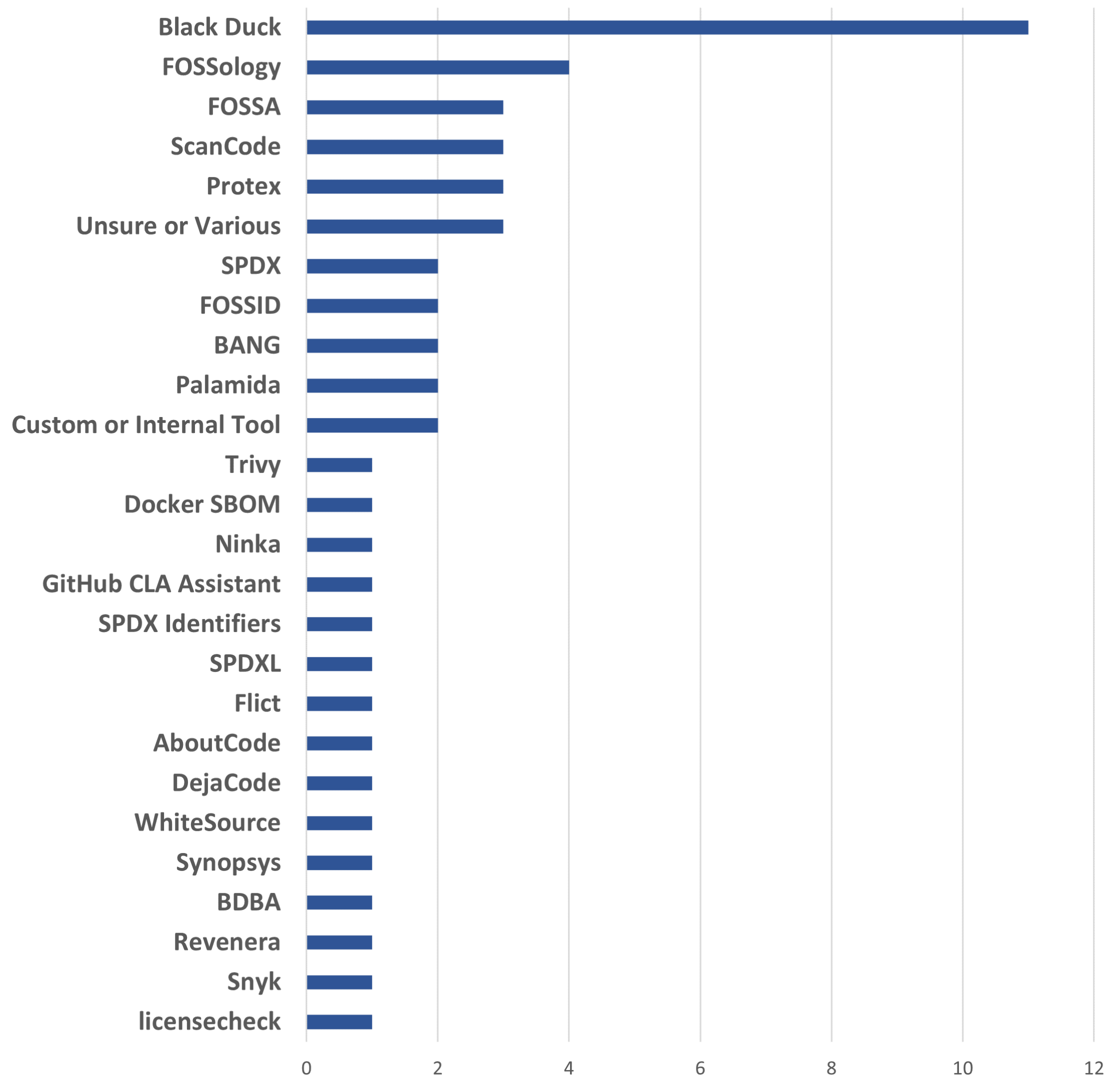

Figure C.2: Tools identified by practitioners

Bibliography

[1] The 3-clause bsd license. https://opensource.org/license/bsd-3-clause/. Accessed: 2023-14-09.

[2] Blackduck software composition analysis. https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html. Accessed: 2023-27-09.

[3] Fossology. https://www.fossology.org/. Accessed: 2023-27-09.

[4] Free software foundation. https://www.fsf.org/. Accessed: 2023-20-09.

[5] Frequently asked questions about the gnu licenses. https://www.gnu.org/licenses/gpl-faq.html. Accessed: 2023-26-09.

[6] Github. https://github.com/. Accessed: 2023-20-09.

[7] Gnu general public license version 3. https://opensource.org/license/gpl-3-0/. Accessed: 2023-14-09.

[8] Gnu lesser general public license version 2.1. https://opensource.org/license/lgpl-2-1/. Accessed: 2023-14-09.

[9] Hacker news. https://news.ycombinator.com/. Accessed: 2023-25-09.

[10] Mit license. https://opensource.org/license/mit/. Accessed: 2023-14-09.

[11] Open source initiative. https://opensource.org/. Accessed: 2023-20-09.

[12] Scancode toolkit. https://github.com/nexB/scancode-toolkit. Accessed: 2023-27-09.

[13] Spdx license expressions (normative). https://spdx.github.io/spdx-spec/v2-draft/SPDX-license-expressions/. Accessed: 2023-03-11.

[14] Spdx license list. https://spdx.org/licenses/. Accessed: 2023-14-09.

[15] U.s. court of appeals for the ninth circuit, lewis galoob toys, inc. v. nintendo of america, inc., 964 f.2d 965., 1992.

[16] Report of license proliferation committee and draft faq. https://opensource.org/proliferation-report/, 7 2006. Accessed: 2023-20-09.

[17] Vlc engine relicensed to lgpl. https://www.videolan.org/press/lgpl-libvlc.html, 12 2011. Accessed: 2023-20-09.

[18] Mysql-mariadb history talk. https://mariadb.org/wp-content/uploads/2019/11/MySQL-MariaDB-story.pdf, 11 2019. Accessed: 2023-27-09.

[19] Copyright registration of computer programs. https://www.copyright.gov/circs/circ61.pdf, 3 2021. Accessed: 2023-25-09.

[20] U.s. code title 17, section 106. https://www.govinfo.gov/app/details/USCODE-2021-title17/USCODE-2021-title17-chap1-sec106/summary, 2021. Accessed: 2023-25-09.

[21] The 2023 am law 100. https://www.law.com/americanlawyer/am-law-100/, 2023.

[22] Hashicorp's licensing change is only the latest challenge to open source. https://thenewstack.io/hashicorp-abandons-open-source-for-business-source-license/, 8 2023. Accessed: 2023-20-09.

[23] Open source security and risk analysis report. `https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html`, 2023. Accessed: 2023-14-09.

[24] Qualtrics. `https://www.qualtrics.com/`, [n.d.]. Accessed: 2023-21-06.

[25] Yasemin Acar, Michael Backes, Sascha Fahl, Doowon Kim, Michelle L Mazurek, and Christian Stransky. You get where you're looking for: The impact of information sources on code security. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 289–305. IEEE, 2016.

[26] Daniel A Almeida, Gail C Murphy, Greg Wilson, and Michael Hoye. Investigating whether and how software developers understand open source software licensing. *Empirical Software Engineering*, 24:211–239, 2019.

[27] Daniel A Almeida, Gail C Murphy, Greg Wilson, and Mike Hoye. Do software developers understand open source licenses? In *2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pages 1–11. IEEE, 2017.

[28] Le An, Ons Mlouki, Foutse Khomh, and Giuliano Antoniol. Stack overflow: A code laundering platform? In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 283–293. IEEE, 2017.

[29] Miriam Ballhausen. Free and open source software licenses explained. *Computer*, 52(6):82–86, 2019.

[30] Sebastian Baltes and Stephan Diehl. Usage and attribution of stack overflow code snippets in github projects. *Empirical Software Engineering*, 24(3):1259–1295, 2019.

[31] Mahak Bandi. All about open source licenses. `https://fossa.com/blog/what-do-open-source-licenses-even-mean/`, 6 2019. Accessed: 2023-24-09.

[32] Knut Blind and Torben Schubert. Estimating the gdp effect of open source software and its complementarities with r&d and patents: evidence and policy implications. *The Journal of Technology Transfer*, pages 1–26, 2023.

[33] Thomas Claburn. Gpl legal battle: Vizio told by judge it will have to answer breach-of-contract claims. `https://www.theregister.com/2022/05/16/vizio_gpl_contract/`, 5 2022. Accessed: 2023-14-09.

[34] Thomas Claburn. John deere urged to surrender source code under gpl. `https://www.theregister.com/2023/03/17/john_deere_sfc_gpl/`, 3 2023. Accessed: 2023-14-09.

[35] Michael Cusumano. Cloud computing and saas as new computing platforms. *Communications of the ACM*, 53(4):27–29, 2010.

[36] Massimiliano Di Penta, Daniel M German, Yann-Gaël Guéhéneuc, and Giuliano Antoniol. An exploratory study of the evolution of software licensing. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering-Volume 1*, pages 145–154, 2010.

[37] Muyue Feng, Weixuan Mao, Zimu Yuan, Yang Xiao, Gu Ban, Wei Wang, Shiyang Wang, Qian Tang, Jiahuan Xu, He Su, et al. Open-source license violations of binary software at large scale. In *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 564–568. IEEE, 2019.

[38] Felix Fischer, Konstantin Böttinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. Stack overflow considered harmful? the impact of copy&paste on android application security. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 121–136. IEEE, 2017.

[39] GR Gangadharan, Vincenzo D'Andrea, Stefano De Paoli, and Michael Weiss. Managing license compliance in free and open source software development. *Information Systems Frontiers*, 14:143–154, 2012.

[40] Daniel German and Massimiliano Di Penta. A method for open source license compliance of java applications. *IEEE software*, 29(3):58–63, 2012.

[41] Daniel M German, Massimiliano Di Penta, and Julius Davies. Understanding and auditing the licensing of open source software distributions. In *2010 IEEE 18th International Conference on Program Comprehension*, pages 84–93. IEEE, 2010.

[42] Daniel M German and Ahmed E Hassan. License integration patterns: Addressing license mismatches in component-based development. In *2009 IEEE 31st international conference on software engineering*, pages 188–198. IEEE, 2009.

[43] Rishab Aiyer Ghosh et al. Economic impact of open source software on innovation and the competitiveness of the information and communication technologies (ict) sector in the eu. *URL https://www. semanticscholar. org/paper/Economic-impact-of-open-source-software-on-and-the-Ghosh/4f0469c3702f5a22176265c72d0d764bc0447774 (accessed 4.18. 20)*, 2007.

[44] Grant Gross. Open-source legal group strikes again on busybox, suing verizon. `https://www.computerworld.com/article/2537947/open-source-legal-group-strikes-again-on-busybox--suing-verizon.html`, 12 2007. Accessed: 2023-14-09.

[45] Robert M. Groves, Floyd J. Jr. Fowler, Mick P. Couyper, James M. Lepkowski, Eleanor Singer, and Roger Tourangeau. *Survey Methodology, 2nd edition*. Wiley, 2009.

[46] Neil Gunningham, Robert A Kagan, and Dorothy Thornton. Social license and environmental protection: why businesses go beyond compliance. *Law & Social Inquiry*, 29(2):307–341, 2004.

[47] Armijn Hemel, Karl Trygve Kalleberg, Rob Vermaas, and Eelco Dolstra. Finding software license violations through binary code clone detection. In *Proceedings of the 8th Working Conference on Mining Software Repositories*, pages 63–72, 2011.

[48] Georgia M Kapitsaki and Georgia Charalambous. Modeling and recommending open source licenses with findosslicense. *IEEE Transactions on Software Engineering*, 47(5):919–935, 2019.

[49] Georgia M Kapitsaki, Frederik Kramer, and Nikolaos D Tselikas. Automating the license compatibility process in open source software with spdx. *Journal of systems and software*, 131:386–401, 2017.

[50] Barbara A. Kitchenham and Shari Lawrence Pfleeger. Principles of survey research part 2: designing a survey. *ACM SIGSOFT Software Engineering Notes*, 27(1):18–20, 2002.

[51] Barbara A. Kitchenham and Shari Lawrence Pfleeger. Principles of survey research: part 3: constructing a survey instrument. *ACM SIGSOFT Software Engineering Notes*, 27(2):20–24, 2002.

[52] Barbara A. Kitchenham and Shari Lawrence Pfleeger. Principles of survey research part 4: questionnaire evaluation. *ACM SIGSOFT Software Engineering Notes*, 27(3):20–23, 2002.

[53] Barbara A. Kitchenham and Shari Lawrence Pfleeger. Principles of survey research: part 5: populations and samples. *ACM SIGSOFT Software Engineering Notes*, 27(5):17–20, 2002.

[54] Barbara A. Kitchenham and Shari Lawrence Pfleeger. Principles of survey research part 6: data analysis. *ACM SIGSOFT Software Engineering Notes*, 28(2):24–27, 2003.

[55] Xiaoyu Liu, LiGuo Huang, Jidong Ge, and Vincent Ng. Predicting licenses for changed source code. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 686–697. IEEE, 2019.

[56] Zhiyou Liu, Zili Zhang, Zhiqiang Wang, Jing Peng, and Sheng Wu. Choosing an open source license based on software dependencies. In *2021 IEEE International Conference on Software Engineering and Artificial Intelligence (SEAI)*, pages 30–36. IEEE, 2021.

[57] Ilyas Saïd Makari, Ahmed Zerouali, and Coen De Roover. Prevalence and evolution of license violations in npm and rubygems dependency networks. In *International Conference on Software and Software Reuse*, pages 85–100. Springer, 2022.

[58] Laura Manor and Junyi Jessy Li. Plain english summarization of contracts. *arXiv preprint arXiv:1906.00424*, 2019.

[59] Heather Meeker. *Open source for business: a practical guide to open source software licensing.* CreateSpace, 2017.

[60] Rômulo Meloca, Gustavo Pinto, Leonardo Baiser, Marco Mattos, Ivanilton Polato, Igor Scaliante Wiese, and Daniel M German. Understanding the usage, impact, and adoption of non-osi approved licenses. In *Proceedings of the 15th International Conference on Mining Software Repositories*, pages 270–280, 2018.

[61] Ron Miller. Terraform fork gets renamed opentofu, and joins linux foundation. `https://techcrunch.com/2023/09/20/terraform-fork-gets-a-`

`new-name-opentofu-and-joins-linux-foundation/`, 9 2023. Accessed: 2023-21-09.

[62] Ons Mlouki, Foutse Khomh, and Giuliano Antoniol. On the detection of licenses violations in the android ecosystem. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 382–392. IEEE, 2016.

[63] Joao Pedro Moraes, Ivanilton Polato, Igor Wiese, Filipe Saraiva, and Gustavo Pinto. From one to hundreds: multi-licensing in the javascript ecosystem. *Empirical Software Engineering*, 26:1–29, 2021.

[64] Nathan Wintersgill, Trevor Stalnaker, Laura A. Heymann, Oscar Chaparro, and Denys Poshyvanyk. Online replication package. `https://github.com/nwintersgill/licensing_issues_study`, 2023.

[65] Philippe Ombredanne. Free and open source software license compliance: tools for software composition analysis. *Computer*, 53(10):105–109, 2020.

[66] Maria Papoutsoglou, Georgia M Kapitsaki, Daniel German, and Lefteris Angelis. An analysis of open source software licensing questions in stack exchange sites. *Journal of Systems and Software*, 183:111113, 2022.

[67] Shari Lawrence Pfleeger and Barbara A. Kitchenham. Principles of survey research: part 1: turning lemons into lemonade. *ACM SIGSOFT Software Engineering Notes*, 26(6):16–18, 2001.

[68] Shi Qiu, Daniel M German, and Katsuro Inoue. Empirical study on dependency-related license violation in the javascript package ecosystem. *Journal of Information Processing*, 29:296–304, 2021.

[69] Chaiyong Ragkhitwetsagul, Jens Krinke, Matheus Paixao, Giuseppe Bianco, and Rocco Oliveto. Toxic code snippets on stack overflow. *IEEE Transactions on Software Engineering*, 47(3):560–581, 2019.

[70] Dirk Riehle and Nikolay Harutyunyan. Open-source license compliance in software supply chains. In *Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability: Communications of NII Shonan Meetings*, pages 83–95. Springer, 2019.

[71] Donna Spencer. *Card sorting: Designing usable categories.* Rosenfeld Media, 2009.

[72] Trevor Stalnaker, Nathan Wintersgill, Oscar Chaparro, Massimiliano Di Penta, Daniel M German, and Denys Poshyvanyk. Boms away! inside the minds of stakeholders: A comprehensive study of bills of materials for software systems. *arXiv preprint arXiv:2309.12206*, 2023.

[73] Amjed Tahir, Aiko Yamashita, Sherlock Licorish, Jens Dietrich, and Steve Counsell. Can you tell me if it smells? a study on how developers discuss code smells and anti-patterns in stack overflow. In *Proceedings of the 22nd International Conference on Evaluation and Assessment in Software Engineering 2018*, pages 68–78, 2018.

[74] Timo Tuunanen, Jussi Koskinen, and Tommi Kärkkäinen. Automated software license analysis. *Automated Software Engineering*, 16:455–490, 2009.

[75] Ashlee Vance. The defenders of free software. `https://www.nytimes.com/2010/09/26/business/26ping.html`, 9 2010. Accessed: 2023-14-09.

[76] Christopher Vendome, Gabriele Bavota, Massimiliano Di Penta, Mario Linares-Vásquez, Daniel German, and Denys Poshyvanyk. License usage and changes: a large-scale study on github. *Empirical Software Engineering*, 22:1537–1577, 2017.

[77] Christopher Vendome, Daniel M German, Massimiliano Di Penta, Gabriele Bavota, Mario Linares-Vásquez, and Denys Poshyvanyk. To distribute or not to distribute? why licensing bugs matter. In *Proceedings of the 40th International Conference on Software Engineering*, pages 268–279, 2018.

[78] Christopher Vendome, Mario Linares-Vásquez, Gabriele Bavota, Massimiliano Di Penta, Daniel German, and Denys Poshyvanyk. Machine learning-based detection of open source license exceptions. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*, pages 118–129. IEEE, 2017.

[79] James Vincent. The lawsuit that could rewrite the rules of ai copyright. `https://www.theverge.com/2022/11/8/23446821/microsoft-openai-github-copilot-class-action-lawsuit-ai-copyright-violation-training-data`, 11 2022. Accessed: 2023-14-09.

[80] Yuhao Wu, Yuki Manabe, Tetsuya Kanda, Daniel M German, and Katsuro Inoue. Analysis of license inconsistency in large collections of open source projects. *Empirical Software Engineering*, 22:1194–1222, 2017.

[81] Weiwei Xu, Hao He, Kai Gao, and Minghui Zhou. Understanding and remediating open-source license incompatibilities in the pypi ecosystem. *arXiv preprint arXiv:2308.05942*, 2023.

[82] Stefano Zacchiroli. A large-scale dataset of (open source) license text variants. In *Proceedings of the 19th International Conference on Mining Software Repositories*, pages 757–761, 2022.