Understanding Industry Requirements for FLOSS Governance Tools

Nikolay Harutyunyan(✉), Andreas Bauer, and Dirk Riehle

Friedrich-Alexander University Erlangen-Nürnberg, 91058 Erlangen, Germany
{nikolay.harutyunyan,andi.bauer}@fau.de,
dirk@riehle.org

Abstract. Almost all software products today incorporate free/libre, and open source software (FLOSS) components. Companies must govern their FLOSS use to avoid potential risks to their intellectual property resulting from the use of FLOSS components. A particular challenge is license compliance. To manage the complexity of license compliance, companies should use tools and well-defined processes to perform these tasks time and cost efficiently. This paper investigates and presents common industry requirements for FLOSS governance tools, followed by an evaluation of the suggested requirements by matching them with the features of existing tools.

We chose 10 industry leading companies through polar theoretical sampling and interviewed their FLOSS governance experts to derive a theory of industry needs and requirements for tooling. We then analyzed the features of a governance tools sample and used this analysis to evaluate two categories of our theory: FLOSS license scanning and FLOSS in product bills of materials. The result is a list of FLOSS governance requirements based on our qualitative study of the industry, evaluated using the existing governance tool features. For higher practical relevance, we cast our theory as a requirements specification for FLOSS governance tools.

Keywords: Open source software · FLOSS · FOSS · Open source governance
FLOSS governance tools · Company requirements for FLOSS tools

1 Introduction

Commercial use of FLOSS is on the rise as more companies realize the benefits of using FLOSS components in their products, going beyond the commonplace use of FLOSS development tools [9, 12, 19, 25, 34, 35]. In 2017 a report by the European Commission estimated that using FLOSS saves the European economy an estimated EUR 114 billion per year directly and up to EUR 399 billion per year overall [11]. However, companies also need to govern and regulate their use of FLOSS components to avoid common threats, such as FLOSS license non-compliance, copyright and patent infringement, that can result in litigation, cease and desist claims or product recalls [2, 33, 37, 39]. In the context of this paper, we define FLOSS governance as the set of processes, best practices and tools employed by companies to use FLOSS components as part of their commercial products while minimizing their risks and maximizing their benefit from such use.

© IFIP International Federation for Information Processing 2018
Published by Springer International Publishing AG 2018. All Rights Reserved
I. Stamelos et al. (Eds.): OSS 2018, IFIP AICT 525, pp. 151–167, 2018.
https://doi.org/10.1007/978-3-319-92375-8_13

FLOSS governance processes and tools can apply to the commercial use, contribution or leadership of FLOSS projects. We limited the scope of this paper only to the commercial use of FLOSS components, intentionally excluding governance considerations of FLOSS contribution or leadership by companies. This is in line with our earlier definition of FLOSS governance. Such focus allowed us to generate an in-depth theory covering the earliest maturity phase of industry involvement with open source that is of highest practical relevance to most companies today and novel to the growing open source research [20].

Despite the practical relevance of the issue, research has been slow to address the use of FLOSS in products. The existing literature is limited to general FLOSS governance research [1, 3, 4], to research of the governance of open source communities and their development practices [26, 28, 36, 40], and to FLOSS license compliance related governance [10, 13–17, 31, 42, 46]. However, past research has not comprehensively addressed FLOSS governance requirements and best practices in industry. A particularly practical aspect of FLOSS governance is its automation through tooling, which ensures increased efficiency and better integration into the development process. Focusing on the specific aspect of FLOSS governance tooling, we addressed this gap by asking the following research question:

RQ: ***What are the core industry requirements for FLOSS governance tools needed to facilitate the use of FLOSS components in commercial products?***

The research method employed is an adaptation of the grounded theory method [5, 6] called the QDAcity RE method for structural domain modeling using qualitative data analysis [23]. We chose this novel, yet promising research method because it enables using qualitative data analysis (QDA) to develop a theory that can be specifically cast as a requirements specification. Answering our research question, we aimed to cast our theory as a list of common industry requirements for FLOSS governance tools. This format is well-understood in the industry and can, therefore, ensure a high practical value of our research results. Data gathering and analysis were performed using formal semi-structured interviews, researcher notes, and materials review. We interviewed 15 FLOSS governance and compliance experts from 10 diverse companies chosen through theoretical sampling of more than 140 companies.

There are few reports on commercial adoption of FLOSS that are cast as lists of requirements focusing on technical and managerial aspects of using FLOSS in proprietary products [46]. However, neither academic nor practitioner literature offers a detailed list of industry requirements for FLOSS governance or its tooling that goes beyond a high-level of abstraction [30]. In this paper, we addressed this research gap with our main contribution – the theory of industry requirements for FLOSS governance tools. Our theory indicated four key categories of FLOSS governance tool requirements in no particular order:

- Tracking and Reuse of FLOSS components
- License Compliance of FLOSS components
- Search and Selection of FLOSS components
- Other requirements (security, education, etc.)

We then broke down each of these categories into detailed requirements and sub-requirements.

To evaluate our theory, we analyzed marketing materials and demos of 6 widely used and representative FLOSS governance tools. We compared the key tool features with our suggested theory and evaluated our proposed requirements confirming many of them. In future publications, we also plan to address other aspects of FLOSS governance in high detail, including industry best practices for FLOSS supply chain management and license compliance.

2 Related Work

The early research on FLOSS governance in companies was part of the broader research on the commercial use of FLOSS development tools and components [1, 20]. In a systematic literature review on FLOSS adoption in industry, Hauge et al. identified only a limited amount of research focusing on FLOSS component selection by companies [7, 8, 22, 45] and knowledge sharing within FLOSS communities [24, 27, 43]. Hauge et al. [20] did not identify any academic studies focused on the actual industry practice of using FLOSS components in products, thus suggesting that further research is needed on this topic. Our literature review confirmed this research gap prompting us to conduct this study of 10 industry-representative companies.

We set our research scope and that of the related work review to the commercial use of FLOSS components in products and industry requirements for FLOSS governance tooling. We explicitly excluded FLOSS governance related to industry contribution to or leadership of FLOSS projects. We did not identify literature explicitly focused on FLOSS governance tool requirements. However, we found indirect references to the topic that we used as a starting point for our research. We derived three key categories of FLOSS governance requirements that can be addressed through tooling:

- Tracking and Reuse of FLOSS components [21, 32, 44]
- License Compliance of FLOSS components [10, 13–17, 31, 42, 46]
- Search and Selection of FLOSS components [7, 8, 22, 41, 45]

Tracking and Reuse. With the growing availability of high-quality FLOSS components, software developers increasingly use FLOSS components in commercial products. FLOSS governance policies in many companies require developers to track and document such FLOSS use [21, 32]. This enables the well-structured management and reuse of FLOSS components that have been added into product software. Umarji et al. [45] suggest using FLOSS governance tools to create and maintain libraries of reusable FLOSS components. Our findings confirm this as one of the industry requirements for FLOSS governance tools.

Other requirements focus on supply chain management [30], automated management of bill of materials [42], maintenance of FLOSS component metadata in product architecture models [38], etc. Our theory confirms and captures these requirements.

License Compliance. Wang and Wang present a number of requirements for industry adoption of FLOSS. Some of these requirements can be translated into industry requirements for FLOSS governance tools. The authors suggest a managerial requirement for license compliance that includes understanding different FLOSS

licenses and documenting their terms [46]. Our theory suggests that industry requires the use of FLOSS governance tools for documenting company interpretation of most common and used FLOSS licenses and their implications. This requirement is also confirmed by industry associations, such as *The Open Source Automation Development Lab eG*, which in 2017 attempted to standardize FLOSS license obligations through checklists and own license describing language that can eventually be used in a FLOSS governance tool [10].

Other industry requirements for compliance tools include automated FLOSS license scanning [14, 15], automated FLOSS code detection in company's codebase and in its supply chain using source code and binary scans [16, 31, 42], checking FLOSS license compatibility when mixing licenses [17] etc. We confirm all these requirements through expert interviews and formalize them in our theory, while recognizing the technological complexity of fulfilling these requirements by the currently existing tooling.

Search and Selection. Umarji et al. [45] surveyed a sample of 69 programmers. Their research suggested that software developers require and use tools for the search and selection of FLOSS components. The majority of the survey respondents said they used general-purpose search engines with some also using project hosting sites and code-specific search engines. Our expert interviews confirmed the requirement for search and selection of FLOSS components. A requirement in our proposed theory formalizes this industry need.

Other industry requirements for search and selection of FLOSS components focus on the automated identification of software families and types of FLOSS communities [41]. Our theory did not confirm the industry requirement for the tool-assisted software family identification, but did confirm the need for the tool-assisted identification and evaluation of FLOSS communities.

Many other requirements are suggested in both academic literature and practitioner white papers. However, in this section, we combined and presented the literature related to only several key requirements due to our narrow scope.

3 Research Method

We conducted a two-step study that consists of:

1. Deriving a theory based on our understanding of key industry requirements for FLOSS governance tools through expert interviews
2. Evaluating our understanding of industry requirements through marketing materials and demos of existing FLOSS governance tools

Our research approach is represented in Fig. 1 and explained below.

For theory building, we conducted 15 interviews with ten industry-leading companies to understand their requirements for FLOSS governance tools.

We employed an adaptation of the grounded theory [5, 6] method called the QDAcity RE method for structural domain modeling using qualitative data analysis [23]. Corbin and Strauss [6] or Charmaz [5] define the grounded theory method as one

that consists of systematic, yet flexible guidelines for collecting and analyzing qualitative data to construct a theory from that data. Kaufmann and Riehle [23] accept this definition, but extend the method to a more structured, traceable and iterative one providing guidelines for data collection, creation and application of a code system. This enabled us to use the QDAcity-RE method for requirements engineering based on our industry expert interviews. The result is a partial theory of industry requirements for FLOSS governance tools cast as a requirements specification.

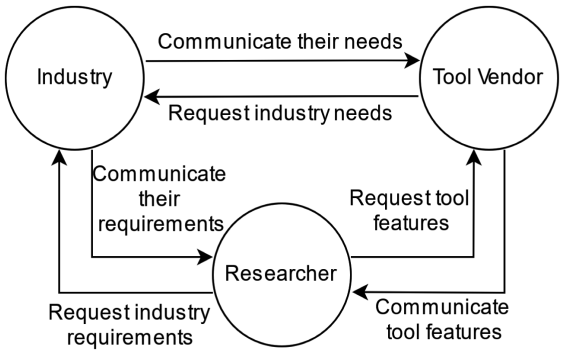

Fig. 1. Theory Building using Industry Requirements and Theory Evaluation using Tool Features

For theory evaluation, we reviewed marketing materials and demos of 6 widely used FLOSS governance tools. We used the QDAcity RE method and qualitative data analysis to derive the common features they offer to meet industry needs for automating FLOSS governance.

Assuming that the tool vendors as a whole understand industry needs and offer tools that address these needs, we compared the common tool features to our partial theory of industry requirements. We evaluated which tool features match the industry requirements in our proposed theory and which ones do not. We used this evaluation to demonstrate that our theory represents the current state of industry requirements for FLOSS governance tools. To the extent that our theory agrees with tool features, we put the work of industry product managers onto a sound scientific base of theory development based on the user's perspective.

3.1 Theoretical Sampling

For theory building, we chose ten companies sampled from our industry network of about 140 companies with advanced FLOSS governance practices. The companies in our sample have advanced understanding of FLOSS governance and use internal and/or external governance tools. We conducted polar theoretical sampling to cover a diverse and representative set of companies. Polar sampling aims to choose companies with highly varying characteristics. We considered diverse dimensions including types of business models, customer types, company size, market position and company maturity. The resulting sample of companies includes small, medium and large companies with both enterprise and retail customers and varying business models. The list of companies and their essential characteristics are presented in Table 1. Company names are anonymized per their request.

Table 1. Theoretical sample of companies

Company	Company domain	By business model	By type of customer	By size (employees)
Company 1	Consulting	SP-OS, SDS	Enterprise	Medium
Company 2	Automotive	SDS	Enterprise	Small
Company 3	Automotive	SDS	Enterprise	Large
Company 4	Enterprise Software	SP-OS	Enterprise, retail	Medium
Company 5	Enterprise Software	SP-CS	Enterprise, retail	Medium
Company 6	Enterprise Software	SP-OS, SP-CS, MC, GT	Enterprise, retail	Large
Company 7	Enterprise Software	SP-OS, MC, GT	Enterprise, retail	Medium
Company 8	FLOSS Foundation	OSF	Enterprise, retail	Small
Company 9	Hardware and Software	OP	Enterprise	Large
Company 10	Legal	MC	Enterprise, government	Large

Legend for Table 1: SDS= Software development service, SP-OS= Software product vendor for open source software,
SP-CS = Software product vendor for closed source software, GT = Governance tool providers, MC = Management consulting,
OSF = Open source foundation, OP = Other products incorporating software

For theory evaluation, we chose 6 widely used and prominent FLOSS governance tools that represent the broader spectrum of FLOSS governance tools [44]. Not all tools compete but have some overlap in their functionalities, like support for license scanning or component repository management. To reduce bias, we made sure that our selection differs in these dimensions:

- By the **license** under which a vendor makes its tool available. The sampling contains tools that are licensed under permissive and copyleft type open source licenses, and proprietary closed source licenses.
- By the **delivery model** of a tool. A critical factor for companies is the ability to choose whether a software tool is available as cloud-based service or can be used on-premise, depending on aspects like costs, customization, and security.
- By the **scannable artifacts**. For scanning of license information, tools can analyze source code or binary artifacts. Scanning of binary artifacts is necessary if the source code of dependent components is not available. In contrast scanning of source code artifacts provide better results.

We also consider other dimensions for the theoretical sampling (maturity of a tool, automation and integration into the development process, and additional audit service by experts), but to offer more depth we focus on the three key dimensions presented above. The list of tools and their key characteristics are presented in Table 2.

Table 2. Sampling of governance tools

Tool	Tool provider	By license	By delivery model	By scannable artifacts
Black Duck Hub	Black Duck Software by Synopsys	Proprietary	Cloud-based	Source and binary code
DejaCode	nexB	Apache 2.0	Cloud-based, on premise	Source and binary code
FOSSology	FOSSology FLOSS project	GPL-2.0	On premise	Source and binary code
FOSSA	FOSSA	Proprietary	Cloud-based, on premise	Source code
OSS-Review-Toolkit	OSS-Review-Toolkit (ORT) FLOSS project	Apache 2.0	On premise	Source code
WhiteSource	WhiteSource Software	Proprietary	Cloud-based, on premise	Source and binary code

3.2 Data Gathering and Analysis

For data gathering, we mainly used semi-structured interviews conducted by one or two researchers with FLOSS governance experts or responsible coworkers from the sampled companies. In seven companies we interviewed one expert, in one company we interviewed two experts, and in two companies we interviewed three experts. In total, we conducted 15 interviews. When possible, we recorded and transcribed the interviews. In three cases we took notes. We also studied additional materials both public and private about these companies and their FLOSS governance practices.

We developed key questions and an interview guideline for the semi-structured interviews and kept them stable, except for few iterative adjustments from company to company, throughout the whole data gathering process. The interviews were exploratory in line with our grounded-theory-based research method.

For data analysis, we followed the QDAcity-RE method performing iterative and incremental qualitative data analysis (QDA) supported by the MaxQDA software. We developed two separate coding systems for the theory building using expert interviews and for the theory evaluation using tool marketing materials and demos.

During the QDA coding process, we iteratively refined the code system. Reaching theoretical saturation [23], the code system became the basis for our theory. Individual codes correspond to low-level tool requirements in our requirements specification. Both for theory building and evaluation, our code systems consist of hierarchical codes. We did not apply the top category codes in our QDA. We followed the QDAcity-RE method's QDA process as follows:

- *Open coding*. We created a basic set of codes from which the hierarchy is built. Open codes are direct annotations of primary materials and link to them for data-theory traceability.
- *Axial coding*. We built a code system by deriving more abstract concepts and categories from open codes, thus developing the axes of the code system.

- *Selective coding*. We applied the codes to the gathered data and chose which codes are important and which are not. We adjusted the coding system by removing the irrelevant codes and by adding the ones that emerged when applying the axial codes.

4 Research Results

This section presents our partial theory of industry requirements for FLOSS governance tools, followed by the evaluation of the suggested theory through feature analysis of existing FLOSS governance tools. Section 4.1 presents our theory cast as a requirements specification for high practical relevance. Section 4.2 presents our evaluation of the theory.

4.1 Theory of Industry Requirements for FLOSS Governance Tools

We limited our scope to FLOSS governance tools related to the commercial use of FLOSS components, explicitly excluding companies' contribution to or leadership of FLOSS projects. We only present the requirements that have been directly derived or inferred from our data, thus excluding the ones that have been presented in the literature, but not confirmed by our industry study. The result is a partial theory that covers the key requirement categories and requirements based on our sample. Analyzing 15 expert interviews, researcher notes and company materials, we derived the following high-level industry requirements for FLOSS governance tools:

1. **Tracking and Reuse of FLOSS components**
   1.1. The tool should help users **identify the use of FLOSS components in their code base**.
   1.2. The tool should help users **report the use of FLOSS components in a product architecture model**.
   1.3. The tool should help users **update FLOSS components and their metadata**.
   1.4. The tool should help users **maintain a bill of materials of the FLOSS components used in a product**.
   1.5. The tool should help users **reuse FLOSS components that have already been used in a product**.

Virtually all companies track their use of FLOSS components in order to efficiently manage FLOSS integration into their products, as well as to enable cost-saving reuse of FLOSS components already used by the company's other developers. Efficient FLOSS component management ensures a company's ability to maintain and produce upon customer request an up-to-date bill of materials. One interview partner mentions this requirement for this use case *(Requirement 1.4)*:

> *"So, we do have tools to keep track of different components or licenses we're using. If you get requests or requirements from customers to provide a list of used [FLOSS] components and licenses, we use this tool to track those and push those requirements into our [development] process." (Company 7)*

Another expert suggests a requirement to enable tracking and reusing FLOSS components *(Requirement 1.5)*:

> *"What we have there at the moment is that [for] half of the company we have essential database or half of the company uses that central database of components and their licenses." (Company 2)*

2. **License Compliance of FLOSS components**
   - 2.1. The tool should help users **interpret open source licenses**.
   - 2.2. The tool should help users **document the identified licenses of the used FLOSS components in the company's open source license repository or license handbook**.
   - 2.3. The tool should help users **find and document the unidentified licenses of the used FLOSS components in the company's open source license repository or license handbook**.
   - 2.4. The tool should help users **approve the use of a FLOSS component in a product based on FLOSS license compliance guidelines**.
   - 2.5. The tool should help users **distribute a product that is compliant with the FLOSS licenses of the FLOSS components used in that product**.

FLOSS license compliance is a central aspect and key tool requirement category to the companies we studied. Companies strive to automate license compliance, license scanning and license management. Some companies employ continuous integration/deployment and thus require appropriate license compliance tools that can be integrated in their development process. Tool requirements for license compliance go on to encompass automated license interpretation, license identification and documentation, etc.

An expert from Company 7 mentions the tool requirement for automating FLOSS license scanning and identification of other FLOSS component metadata *(Requirement 2.2)*, as well as the requirement for automating component approval *(Requirement 2.4)*:

> *"We have a full toolset that goes through and scans the code, that pulls out all the license information, the authorship [copyright] information, and runs that through our process for verification, for compliance, for compatibility and so forth." (Company 7)*

Another expert talks about the need to find and document the unidentified FLOSS components and licenses *(Requirement 2.3)*:

> *"We need this [license scanning] tool to re-check if any of the developers are not handling [FLOSS components] in the way [the management] wants, to better do it because we have no possibility to check it in a clear way if you have no tool." (Company 3)*

3. **Search and Selection of FLOSS components**
   - 3.1. The tool should help users **search for FLOSS components**.
   - 3.2. The tool should help users **select best FLOSS components**.
   - 3.3. The tool should help users **estimate the cost of using an FLOSS component**.

Companies need to FLOSS governance tools to efficiently search and select the right FLOSS components, which translates into tool requirements on evaluating

different component candidates and estimating the cost of their usage. One interviewee talks about the role of tools in FLOSS component selection process *(Requirement 3.2)*:

> *"When you move on from a strategic decision to component selection like with components of open source projects to be used, then we have a process that we require the projects to name all the open source components to assess that they want to use, that they assess the license, that they check the license, and that they document that and that again this assessment is communicated to upper management and signed off that." (Company 2)*

4. **Other requirements**
   4.1. The tool should help users **detect and prevent security vulnerabilities in product's FLOSS components**.
   4.2. The tool should help users **document and communicate company's FLOSS governance strategy, policies and best practices**.
   4.3. The tool should help users get **training on FLOSS governance and compliance when using open source software in products and contributing to open source projects**.

The detailed subcategories of requirements for Tracking and Reuse of FLOSS components are demonstrated in Table 3. The detailed subcategories of requirements for License Compliance of FLOSS components are demonstrated in Table 4. The detailed subcategories of requirements for Search and Selection of FLOSS components are demonstrated in Table 5.

4.2 Evaluation

This section presents the evaluation of our suggested theory using the feature analysis of existing FLOSS governance tools. We analyzed marketing materials and demos of six widely used FLOSS governance tools. The analysis resulted in the following list of common key features related to FLOSS use in products:

- *Component Tracking & Reporting*: support for bill of materials, component inventory, knowledge base (external inventory), license obligation reporting, and commonly accepted data exchange standard support;
- *Scanning/License Checking*: support for licenses identification, copyright identification, code origin identification, and license management;
- *Policies*: support for applying/ensuring FLOSS policies;
- *Security*: support for security vulnerability detection;
- *Development Integration & Automation*: support for integration into continuous integration and deployment.

To ensure the depth of evaluation, we focus on two main requirement categories: **Tracking and Reuse of FLOSS components** and **License Compliance of FLOSS components**. We chose these categories because these requirements are fundamental to any software company according to the analysis of the industry interviews and tools support of these requirements as base functionalities.

Table 3. Tracking and Reuse of FLOSS components requirements

1. The tool should help users **identify the use of FLOSS components in their code base**.
   a. The tool should allow reading in an existing code base.
   b. The tool should allow automated finding of open source licenses in an existing code base.
   c. The tool should allow automated finding of open source software checked-in and used by a company developer.
   d. The tool should allow automated finding of open source software not checked-in, but used by a company developer.
   e. The tool should allow automated finding of open source software that is part of the supplied proprietary software using commonly accepted data exchange standards (such as SPDX).
   f. The tool should allow automated finding of open source software that is part of the supplied proprietary software using binary or source code scanning.

2. The tool should help users **report the use of FLOSS components in a product architecture model**.
   a. The tool should allow creating a product architecture model to systematically record use of FLOSS components, their metadata and component dependencies.
   b. The tool should allow manual recording of metadata of the used FLOSS components.
   c. The tool should allow confirming the metadata of FLOSS components identified automatically.
   d. The tool should allow modifying the metadata of FLOSS components identified automatically.
   e. The tool should allow removing the metadata of FLOSS components identified automatically.
   f. The tool should allow automated reporting of a newly used FLOSS component within the build process and/or continuous integration process.
   g. The tool should allow reporting undeclared use of FLOSS components and their metadata.

3. The tool should help users **update FLOSS components and their metadata**.
   a. The tool should allow automated updates of FLOSS components to their newest available versions.
   b. The tool should allow to back up the current versions of FLOSS components before updating them.
   c. The tool should allow automated identification of changed metadata including FLOSS component license and copyright information.
   d. The tool should allow automated history recording of FLOSS components and their metadata.

4. The tool should help users **maintain bill of materials of the FLOSS components used in a product**.
   a. The tool should allow creating a formal bill of material using a commonly accepted data exchange standard (such as SPDX).
   b. The tool should allow automated generation of a formal bill of materials using company's product architecture model.
   c. The tool should allow developers to add identified and reported metadata on used FLOSS components into the formal bill of materials.
   d. The tool should allow developers to update the formal bill of materials.
   e. The tool should allow automated generation of a bill of materials instance in a structured textual format.
   f. The tool should allow automated generation of a bill of materials instance in a commonly accepted data exchange standard (such as SPDX) format.

5. The tool should help users **reuse FLOSS components that have already been used in a product**.
   a. The tool should allow creating a centralized and company-wide accessible FLOSS component repository.
   b. The tool should allow automated adding of FLOSS components and their metadata into the repository using the product architecture model.
   c. The tool should allow automated updating of FLOSS components repository using the product architecture model.
   d. The tool should allow all company developers to access the FLOSS components repository.
   e. The tool should allow searching in the FLOSS component repository.
   f. The tool should allow finding the company developers who used an FLOSS component from the repository.

Tracking and Reuse of FLOSS components. The identification of FLOSS components and their licenses in a given software product or component is a core functionality of all sampled tools. All the high-level requirements of the category 1 in the proposed theory are matched by the features of the sampled tools. For example, Black Duck Software enables its users to identify the used FLOSS components *(Requirement 1.1)* in both the source code and in binaries (with lesser precision):

"[Black Duck Hub enables to] fully discover all open source in your code" (Black Duck Hub)

Table 4. License Compliance of FLOSS components requirements

1. The tool should help users **interpret open source licenses**.
   a. The tool should allow user to document open source license interpretations using a formal language or notation supported by the tool.
   b. The tool should provide automated standard interpretation of the most common FLOSS licenses in company's license repository or license handbook.
   c. The tool should allow users to modify license interpretation of the most common FLOSS licenses in company's license repository or license handbook.
   d. The tool should allow users to add license interpretation of the FLOSS licenses of the used FLOSS components to company's license repository or license handbook.
   e. The tool should allow users to change license interpretation in the license repository or license handbook.
   f. The tool should allow developers to request license interpretation of a FLOSS license of an FLOSS component s/he wants to use in a product.
   g. The tool should allow open source program office to discuss license interpretation requests.
   h. The tool should allow open source program office to fulfill license interpretation requests.

2. The tool should help users **document the identified licenses of the used FLOSS components in the company's open source license repository or license handbook**.
   a. The tool should allow creating an open source license repository.
   b. The tool should allow developers, lawyers and managers to read the open source license repository.
   c. The tool should allow automated inventorying of known open source licenses from the product architecture model.
   d. The tool should allow users to add new open source licenses into the open source license repository.
   e. The tool should allow users to remove obsolete open source licenses from the open source license repository.
   f. The tool should support the commonly accepted data exchange standards (such as SPDX).
   g. The tool should allow users to search open source license information in the open source license.

3. The tool should help users **find and document the unidentified licenses of the used FLOSS components in company's open source license repository or license handbook**.
   a. The tool should allow software package scanning to find the open source licenses unidentified previously through product architecture model.
   b. The tool should allow source code scanning for the internally developed code to find the origin of used, but unidentified open source code and its license.
   c. The tool should allow source code scanning for the FLOSS components taken from FLOSS projects to find the origin of used, but unidentified open source code and its license.
   d. The tool should allow binary scanning for the FLOSS components that are part of the supplied proprietary software components to find the origin of used, but unidentified open source code and its license.
   e. The tool should allow automated inventorying of the open source licenses identified because of binary and source code scanning.
   f. The tool should allow manual changing the automatically identified open source licenses.
   g. The tool should allow removing the automatically identified open source licenses.
   h. The tool should support binary and source code scanning integration into the build process and/or continuous integration process.
   i. The tool should allow finding and documenting copyright notices, export restriction information and other compliance-related metadata for FLOSS components used in a product.

4. The tool should help users **approve the use of a FLOSS component in a product based on FLOSS license compliance guidelines**.
   a. The tool should allow creating white lists of company-approved FLOSS licenses according to company policy.
   b. The tool should allow creating black lists of company-blocked FLOSS licenses according to company policy.
   c. The tool should allow updating white and black lists of FLOSS licenses.
   d. The tool should allow creating license interpretation-based rules for automated recommendation on component use approval according to company policy.
   e. The tool should allow developers to request approval of FLOSS components with previously unassessed licenses.
   f. The tool should allow lawyers to approve or block use of FLOSS components due to license incompatibility with company policy.
   g. The tool should allow automated recording of FLOSS license approval decisions in company's open source license repository.

5. The tool should help users **distribute a product that is compliant with the FLOSS licenses of the FLOSS components used in that product**.
   a. The tool should allow automated generating of FLOSS license obligations for each product using product architecture model and open source license repository.
   b. The tool should allow automated assignment of tasks that will ensure compliance with FLOSS license obligations.
   c. The tool should allow automated audit of product's bill of materials before distribution.
   d. The tool should allow manual audit of product's bill of materials before distribution.
   e. The tool should allow adjusting product's bill of materials before distribution.

Table 5. Search and Selection of FLOSS components requirements

1. The tool should help users **search for FLOSS components**. a. The tool should allow automated search of available FLOSS components using publicly available data. b. The tool should allow automated comparison of available FLOSS components using publicly available data.
2. The tool should help users **select best FLOSS components**. a. The tool should allow automated health assessment of open source communities using publicly available data. b. The tool should allow automated maturity assessment of open source communities using publicly available data. c. The tool should allow automated corporate dependence assessment of open source communities using publicly available data. d. The tool should allow automated maturity assessment of open source communities using publicly available data. e. The tool should allow automated responsiveness assessment of open source communities using publicly available data.
3. The tool should help users **estimate the cost of using an FLOSS component**. a. The tool should allow automated cost estimation of FLOSS component integration and maintenance in a product. b. The tool should allow automated risk assessment of FLOSS community discontinuing its development of the FLOSS component and automated cost estimation of internal maintenance of the FLOSS component. c. The tool should allow users semi-automated estimation of the benefit of using an FLOSS component compared to proprietary and in-house development alternatives.

FOSSA helps explore and report relationships between modules incl. the open source ones *(Requirement 1.2)*:

> *"[FOSSA allows its user to] explore relationships between modules and if/how dependencies are included in your build" (FOSSA)*

Black Duck Hub also has features for BOM maintenance *(Requirement 1.4)* and for FLOSS component reuse *(Requirement 1.5)*:

> *"We provide a license obligation report, including an easily consumable bill of materials (BOM) that you can deliver to your customers and/or internal stakeholders." (Black Duck Hub)*

> *"[Black Duck Hub enables to] eliminate uncertainty and promote reuse [of FLOSS]" (Black Duck Hub)*

However, not all detailed (low-level) requirements from the proposed theory are supported by existing tool features. *Requirement 1.1.d*, for example requires tools to allow automated finding of open source software not checked-in but used by a company developer. This requirement is not entirely supported by any of the studied tool because of its technological complexity.

License Compliance of FLOSS components. All the studied tools support FLOSS license compliance features. They fulfill offer fulfilling requirements, such as license interpretation, license identification and documentation, FLOSS component approval etc.

FOSSology covers several requirements related to FLOSS license compliance *(Requirement 2.2, 2.3)* [18]:

> *"FOSSology is an open source license compliance software system and toolkit. As a toolkit you can run license, copyright and export control scans from the command line. As a system, a database and web* UI *are provided to give you a compliance workflow. License, copyright and export scanners are tools available to help with your compliance activities." (FOSSology)*

However, none of our studied tools completely fulfill some of the following low-level requirements: *Requirement 2.1.b* (automated standard interpretation of common FLOSS licenses), *Requirement 2.3.h* (automated license checking within continuous integration), *Requirement 2.5.b* (automated assignment of FLOSS compliance tasks), *Requirement 2.5.c* (automated audit of product's bill of materials before distribution). One reason is the complex computational nature of the complete automation of compliance tasks. An empirical study by German et al. [16] showed that a deeper understanding of licensing issues requires human expertise, which limits the automation of some license compliance tasks. Moreover, most companies don't allow complete automation of compliance as they require a human actor to be responsible for legal matters, even if they use semi-automated tooling.

Our limited evaluation demonstrates that the high-level requirements of our theory do match the features offered by industry leading FLOSS governance tools. The evaluation shows that existing tools satisfy most of the low-level requirements by the industry, but not others, such as requirements of complete automation.

5 Discussion

Our main contribution is the requirements specification presented in Sect. 4.1 and its evaluation in Sect. 4.2.

We recognize that our research results are limited, but novel and practice relevant. They present only a partial theory on the issue. However, we lay groundwork for future studies into FLOSS governance tool requirements, that will hopefully expand our requirements specification theory. Our work leads us to propose the following research questions for future research:

RQ1: *What are other detailed FLOSS governance tool requirements beyond Tracking and Reuse of FLOSS components, License Compliance of FLOSS components and Search and Selection of FLOSS components?*
RQ2: *How can FLOSS governance tool requirement theories be better evaluated or validated?*
RQ3: *How to engineer FLOSS governance tool requirements of the future addressing missing features and industry needs before companies become aware of them?*

6 Research Limitations

The study faces several limitations including those to internal validity and to external validity:

Internal validity. Qualitative data research realized by one researcher has inherent subjectivity and bias. Even though we followed the research method constructs carefully, there is bias associated with method interpretation and application to our specific context. To address this limitation, we had a second coder analyze our data and

improved our original QDA coding with that of the second coder. The high inter-coder agreement between the original coding and the second coder coding suggests an adequate quality of our code system and by extension an adequate quality of the derived theory [29].

External validity. The resulting theory is based on the data gathered from the experts of the ten companies we interviewed. We cannot claim broad generalizability of the findings, even though we followed a careful theoretical sampling to ensure the applicability of our results. This limitation can be tested with further validation studies.

7 Conclusion

This paper presents a study of ten industry companies with advanced FLOSS governance practices. Our study concluded in a partial theory of FLOSS governance tool requirements by the industry. Also, we provide a detailed hierarchical list of these industry relevant requirements. As such it offers unique insight into industry understanding of FLOSS governance tools and their expectations from them, alongside existing tools and their features.

The data gathered through semi-structured interviews and materials collection was analyzed using the novel adoption of grounded theory method – the QDAcity-RE method. We cast our theory as a requirements specification making it applicable and practice relevant to the companies willing to employ these requirements. Finally, we evaluated our findings using six industry leading FLOSS governance tools and the analysis of their features matched with the requirements of the suggested theory.

The study of the missing features of existing tools is out of scope of this paper but it can be a valuable part of further research. Further research can also focus on the reasons why tool providers do not fulfill the unsatisfied requirements of our theory (e.g. full automation of compliance) and how such problems can be solved.

Acknowledgments. We would like to thank Hannes Dohrn, Michael Dorner, Maximilian Capraro, Andreas Kaufmann and Shushanik Hakobyan for their generous feedback that helped us improve our paper. We would also like to thank our industry partners that provided their valuable time and expertise for this research project.

References

1. Aksulu, A., Wade, M.: A comprehensive review and synthesis of open source research. J. Assoc. Inf. Syst. **11**(11), 576 (2010)
2. Black Duck Software: 2017 Open Source Security and risk analysis. Center for Open Source Research & Innovation. In: (self-published white paper) (2017)
3. Bonaccorsi, A., Rossi, C.: Why open source software can succeed. Res. Policy **32**(7), 1243–1258 (2003)
4. Capra, E., Francalanci, C., Merlo, F.: An empirical study on the relationship between software design quality, development effort and governance in open source projects. IEEE Trans. Softw. Eng. **34**(6), 765–782 (2008)
5. Charmaz, K.: Constructing Grounded Theory. Sage, Thousand Oaks (2014)

6. Corbin, J., Strauss, A.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage Publications, Thousand Oaks (2014)
7. Cruz, D., Wieland, T., Ziegler, A.: Evaluation criteria for free/open source software products based on project analysis. Softw. Process Improv. Pract. **11**(2), 107–122 (2006)
8. Deprez, J.-C., Alexandre, S.: Comparing assessment methodologies for free/open source software: OpenBRR and QSOS. In: Jedlitschka, A., Salo, O. (eds.) PROFES 2008. LNCS, vol. 5089, pp. 189–203. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69566-0_17
9. Deshpande, A., Riehle, D.: The total growth of open source. In: Russo, B., Damiani, E., Hissam, S., Lundell, B., Succi, G. (eds.) OSS 2008. ITIFIP, vol. 275, pp. 197–209. Springer, Boston, MA (2008). https://doi.org/10.1007/978-0-387-09684-1_16
10. Emde, C., Jaeger, T.: Open source license obligations checklists (version 5). In: Open Source Automation Development Lab (self-published white paper) (2017)
11. European Commission: The economic and social impact of software & services on competitiveness and innovation (SMART 2015/0015). Publications Office of the European Union, Luxembourg, pp. 197–198 (2017)
12. Fitzgerald, B.: The transformation of open source software. MIS Q. **30**(3), 587–598 (2006)
13. Gangadharan, G.R., De Paoli, S., D'Andrea, V., Weiss, M.: License compliance issues in free and open source software. In: MCIS 2008 Proceedings, vol. 2 (2008)
14. Gangadharan, G.R., D'andrea, V., De Paoli, S., Weiss, M.: Managing license compliance in free and open source software development. Inf. Syst. Front. **14**(2), 143–154 (2012)
15. German, D.M., Hassan, A.E.: License integration patterns: Addressing license mismatches in component-based development. In: Proceedings of the 31st International Conference on Software Engineering, pp. 188–198. IEEE Computer Society, May 2009
16. German, D.M., Di Penta, M., Davies, J.: Understanding and auditing the licensing of open source software distributions. In: 2010 IEEE 18th International Conference on Program Comprehension (ICPC), pp. 84–93. IEEE, June 2010
17. German, D.M., Manabe, Y., Inoue, K.: A sentence-matching method for automatic license identification of source code files. In: Proceedings of the IEEE/ACM International Conference on Automated Software Engineering, pp. 437–446. ACM, September 2010
18. Gobeille, R.: The fossology project. In: Proceedings of the 2008 International Working Conference on Mining Software Repositories, pp. 47–50. ACM, May 2008
19. Hammond, J., Santinelli, P., Billings, J.J., Ledingham, B.: The tenth annual future of open source survey. In: Black Duck Software (2016). (self-published presentation)
20. Hauge, Ø., Ayala, C., Conradi, R.: Adoption of open source software in software-intensive organizations–A systematic literature review. Inf. Softw. Technol. **52**(11), 1133–1154 (2010)
21. Helmreich, M.: Best practices of adopting open source software in closed source software products. In: (Doctoral dissertation, Diplomarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg) (2011)
22. Hummel, O., Janjic, W., Atkinson, C.: Code conjurer: pulling reusable software out of thin air. IEEE Softw. **25**(5), 45–52 (2008)
23. Kaufmann, A., Riehle, D.: The QDAcity-RE method for structural domain modeling using qualitative data analysis. Requirements Eng. 1–18 (2017)
24. von Krogh, G., Spaeth, S., Haefliger, S.: Knowledge reuse in open source software: An exploratory study of 15 open source projects. In: 2005 Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS 2005 p. 198b. IEEE, January 2005
25. Von Krogh, G., Von Hippel, E.: The promise of research on open source software. Manage. Sci. **52**(7), 975–983 (2006)

26. De Laat, P.B.: Governance of open source software: state of the art. J. Manage. Governance **11**(2), 165–177 (2007)
27. Lakhani, K.R., Von Hippel, E.: How open source software works:"free" user-to-user assistance. Res. Policy **32**(6), 923–943 (2003)
28. Lattemann, C., Stieglitz, S.: Framework for governance in open source communities. In: 2005 Proceedings of the 38th Annual Hawaii International Conference on System Sciences, HICSS 2005, p. 192a. IEEE, January 2005
29. Lombard, M., Snyder-Duch, J., Bracken, C.C.: Content analysis in mass communication: assessment and reporting of intercoder reliability. Hum. Commun. Res. **28**(4), 587–604 (2002)
30. OpenChain Specification (2018). https://www.openchainproject.org/spec
31. Di Penta, M., German, D.M., Antoniol, G.: Identifying licensing of jar archives using a code-search approach. In: 2010 7th IEEE Working Conference on Mining Software Repositories (MSR), pp. 151–160. IEEE, May 2010
32. Popp, K.M.: Best practices for commercial use of open source software. In: Business Models, Processes and Tools for Managing Open Source Software. BoD–Books on Demand (2015)
33. Radcliffe, M., Odence, P.: The 2017 open source year in review. Black Duck Software, DLA Piper. (self-published presentation) (2017)
34. Riehle, D.: The economic motivation of open source software: stakeholder perspectives. Computer **40**(4), 25–32 (2007)
35. Riehle, D.: The commercial open source business model. In: Nelson, M.L., Shaw, M.J., Strader, T.J. (eds.) AMCIS 2009. LNBIP, vol. 36, pp. 18–30. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03132-8_2
36. Riehle, D.: Controlling and steering open source projects. IEEE Comput. **44**(7), 93–96 (2011)
37. Riehle, D., Lempetzeder, B.: Erfolgsmethoden der Open-Source-Governance und-Compliance. In: Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) (2014)
38. Riehle, D., Harutyunyan, N.: License clearance in software product governance. In: NII Shonan (2017)
39. Ruffin, C., Ebert, C.: Using open source software in product development: a primer. IEEE Softw. **21**(1), 82–86 (2004)
40. Sadowski, B.M., Sadowski-Rasters, G., Duysters, G.: Transition of governance in a mature open software source community: Evidence from the debian case. Inf. Econ. Policy **20**(4), 323–332 (2008)
41. Semeteys, R.: Method for qualification and selection of open source software. In: Open Source Business Resource, May 2008
42. Software Package Data Exchange (SPDX) (2018). https://spdx.org/
43. Sowe, S.K., Stamelos, I., Angelis, L.: Understanding knowledge sharing activities in free/open source software projects: an empirical study. J. Syst. Softw. **81**(3), 431–446 (2008)
44. Tools for Managing Open Source Programs (2018). https://www.linuxfoundation.org/tools-managing-open-source-programs/
45. Umarji, M., Sim, S.E., Lopes, C.: Archetypal internet-scale source code searching. In: Russo, B., Damiani, E., Hissam, S., Lundell, B., Succi, G. (eds.) OSS 2008. ITIFIP, vol. 275, pp. 257–263. Springer, Boston, MA (2008). https://doi.org/10.1007/978-0-387-09684-1_21
46. Wang, H., Wang, C.: Open source software adoption: a status report. IEEE Softw. **18**(2), 90–95 (2001)