Understanding Industry Requirements for FLOSS Governance Tools

Andreas Bauer, Nikolay Harutyunyan, Dirk Riehle
Friedrich-Alexander University Erlangen-Nürnberg

Bitkom Forum Open Source – 2018-09-18

Motivation

- Commercial use of FLOSS (Free/Libre and Open Source Software) is on the rise
- European Commission estimated that using FLOSS saves the European economy an estimated EUR 114 billion per year
- **BUT** companies need to govern their use of FLOSS components to avoid potential threats
- **SO** they use tools for FLOSS governance and compliance
- **BUT** there is no common understanding of industry requirements for such tools
- **SO** we collected and studied these requirements that can benefit the industry and contribute to the open source research

What are the main industry requirements for FLOSS governance tools needed to facilitate the use of FLOSS components in commercial products?

Understanding Industry Requirements for FLOSS Governance Tools

Nikolay Harutyunyan, Andreas Bauer, Dirk Riehle

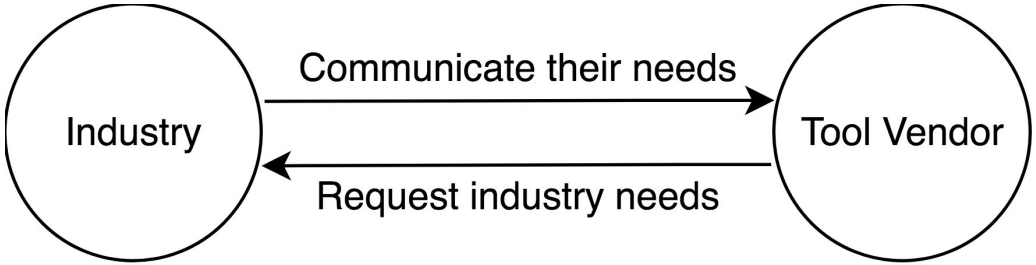
Friedrich-Alexander University Erlangen-Nürnberg, 91058 Erlangen, Germany
{nikolay.harutyunyan,andi.bauer}@fau.de, dirk@riehle.org

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.

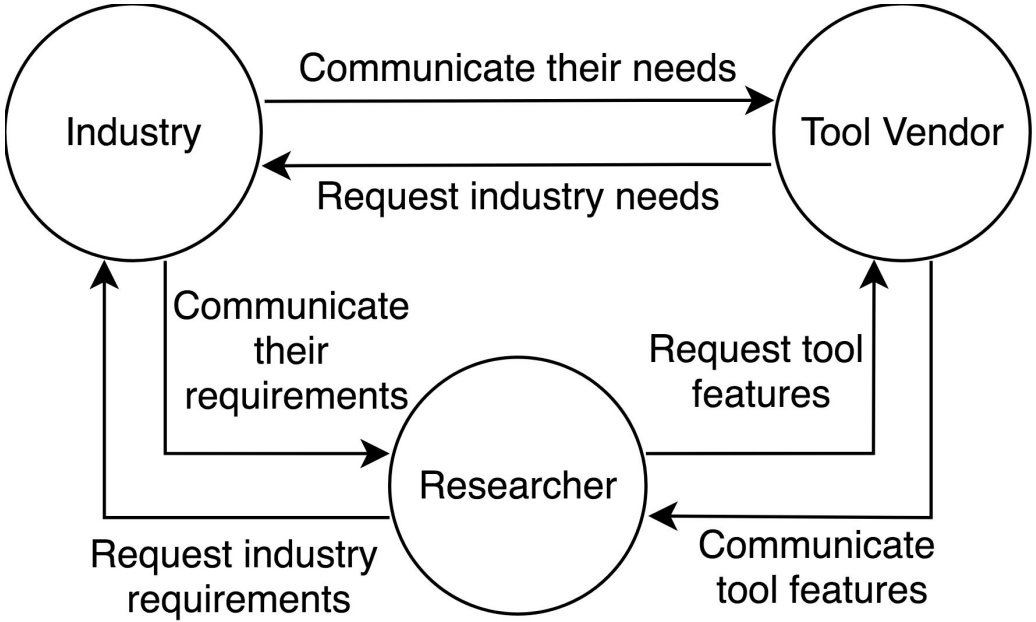
We chose 10 industry leading companies through polar theoretical sampling and interviewed their FLOSS governance experts to derive a theory of industry needs and requirements for tooling. We then analyzed the features of a governance tools sample and used this analysis to evaluate two categories of our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

Harutyunyan, N., Bauer, A., & Riehle, D. (2018). Understanding Industry Requirements for FLOSS Governance Tools (pp. 151–167). Springer, Cham. https://doi.org/10.1007/978-3-319-92375-8_13

Research Context



Research Context



Data Gathering

- **Data**: 15 expert interviews in 10 companies chosen through theoretical sampling of 140 companies with advanced FLOSS governance practices
- **Data Analysis**: Qualitative Data Analysis (QDA) following QDAcity-RE* method

*Kaufmann, A., & Riehle, D. (2017). The QDAcity-RE method for structural domain modeling using qualitative data analysis. Requirements Engineering, 1-18.

Data Gathering: 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

Table Legend: 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

Data Gathering: Tools

Tool	Tool provider	By license	By delivery model	By scannable artifacts
Black Duck Hub	BDS 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

Findings

Main Requirements for FLOSS Gov. Tools

1. **Tracking and Reuse** of FLOSS components
2. **License Compliance** of FLOSS components
3. **Search and Selection** of FLOSS components
4. Other requirements

1. Tracking and Reuse of FLOSS Components

The tool should help users…

1.1. ….**identify** the use of FLOSS components in their code base.

1. Tracking and Reuse of FLOSS Components

The tool should help users…

1.1. ….**identify** the use of FLOSS components in their code base.
1.2. ….**report** the use of FLOSS components in a product architecture model.

1. Tracking and Reuse of FLOSS Components

The tool should help users…

1.1. ….**identify** the use of FLOSS components in their code base.
1.2. ….**report** the use of FLOSS components in a product architecture model.
1.3. ….**update** FLOSS components and their metadata.
1.4. ….**maintain** a bill of materials of the FLOSS components used in a product.
1.5. ….**reuse** FLOSS components that have already been used in a product.

1. Tracking and Reuse of FLOSS Components

1.4. The tool should help users maintain a bill of materials of the FLOSS components used in a product.

”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)

1. Tracking and Reuse of FLOSS Components

1.5. The tool should help users reuse FLOSS components that have already been used in a product.

1.5.a) The tool should allow **creating** a centralized and company-wide accessible FLOSS component repository.

1.5.b) ...automated **adding** of FLOSS components and their metadata into the repository using the product architecture model.

1.5.c) ...automated **updating** of FLOSS components repository using the product architecture model.

1.5.d) ...all company developers to **access** the FLOSS components repository.

1.5.e) ...**searching** in the FLOSS component repository.

1.5.f) ...**finding** the company developers who used an FLOSS component from the repository.

Requirements Unfulfilled by Tools

Not fulfilled by tools:

- Automated standard interpretation of common FLOSS licenses.
- Automated license checking within continuous integration.
- Automated assignment of FLOSS compliance tasks.
- Automated audit of product's bill-of-materials before distribution.

Assumption:
A deeper understanding of licensing issues requires human expertise, which limits the automation of some license compliance task. [German et al. 2010]

"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 4. The detailed subcategories of requirements for License Compliance of FLOSS components are demonstrated in Table 5. The detailed subcategories of requirements for Search and Selection of FLOSS components are demonstrated in Table 6.

Table 4. 1. 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.

Table 5. 2. 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 6. 3. 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.

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 (CI/CD) process.

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.

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):

Conclusion & Future Research

Conclusion

- Tool vendors understand industry relevant requirements.
- We identified 3 main categories of requirements.
- Only a small part of the requirements can not be fulfilled by the tools.
- The findings can become groundwork for future studies.

Future Research

This work is a groundwork for future studies on FLOSS governance tool requirements.

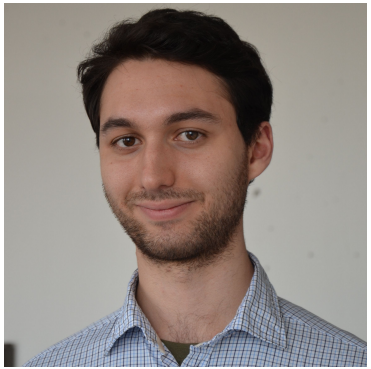
- **RQ1**: What are other detailed FLOSS governance tool requirements beyond Tracking and Reuse of FLOSS components, License Compliance of FLOSS components and Search and Selection of FLOSS components?
- **RQ2**: How can FLOSS governance tool requirement theories be better evaluated or validated?
- **RQ3**: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

Thank you! Questions?

Andreas Bauer, M.Sc.

andi.bauer@fau.de

Researcher and PhD candidate for open source governance and compliance tooling.



Nikolay Harutyunyan, M.Sc.

nikolay.harutyunyan@fau.de

Researcher and PhD candidate for open source governance.

Prof. Dr. Dirk Riehle, M.B.A.

dirk.riehle@fau.de

Professor and head of Open Source Research Group at Friedrich-Alexander-Universität Erlangen-Nürnberg.