Open Source Software Governance: Distilling and Applying Industry Best Practices

Nikolay Harutyunyan

Abstract Modern software architectures are becoming increasingly complex and interdependent. The days of exclusive in-house software development by companies are over. A key force contributing to this shift is the abundant use of open source frameworks, components, and libraries in software development. Over 90% of all software products include open source components. Being efficient, robust, and affordable, they often cover the non-differentiating product requirements companies have. However, the uncontrolled use of open source software in products comes with legal, engineering, and business risks stemming from incorrect software licensing, copyright issues, and supply chain vulnerabilities. While recognized by a handful of companies, this topic remains largely ignored by the industry and little studied by the academia. To address this relevant and novel topic, we undertook a 3-year research project into open source governance in companies, which resulted in a doctoral dissertation. The key results of our work include a theory of industry best practices, where we captured how more than 20 experts from 15 companies worldwide govern their corporate use of open source software. Acknowledging the broad industry relevance of our topic, we developed a handbook for open source governance that enabled practitioners from various domains to apply our findings in their companies. We conducted three evaluation case studies, where more than 40 employees at three Germany-based multinational companies applied our proposed best practices. This chapter presents the highlights of building and implementing the open source governance handbook.

1 Introduction

Traditionally, companies tended to develop large parts of their software products internally with the occasional outsourcing or supplier code. However, modern software development requires the use of free/libre and open source software

N. Harutyunyan (✉)
Friedrich-Alexander-Universität, Erlangen-Nürnberg, Germany
e-mail: nikolay.harutyunyan@fau.de

M. Felderer et al. (eds.), *Ernst Denert Award for Software Engineering 2020*,
https://doi.org/10.1007/978-3-030-83128-8_5

(FLOSS) components. Major software infrastructures are built using Linux servers; many projects rely on open source libraries in Python, web browsers, and database management systems; and much more are critically dependent on the use of open source software. Moreover, many successful companies embrace open source software development and contribute their own software as open source projects including TensorFlow,[1] a library for machine learning developed by Google; React,[2] a front end library for building user interfaces or UI components; Android; Angular; and Chromium just to name a few. While these and many other FLOSS components are used daily by companies, few of them know about the associated risks of such unstructured use.

When not properly managed, incorporating open source software in a company product can have unforeseen legal, technical, and business consequences. A common example is the unintended use of an open source component licensed under a GPL (GNU General Public License) license,[3] which requires the user of the code to in turn open source their software under the same license. If an unaware developer uses a piece of GPL-licensed code in a company product, the company will eventually be forced to either open source the whole product (which might not make business sense) or replace the incorporated open source component. Over the years, companies have been dealing with copyright and licensing violations [1, 2], issues with complex licensing [3, 4], technical issues due to the dependency on other open source projects [5, 6] and low quality documentation [7, 8], and other risks [9–11].

Another major challenge of the uncontrolled FLOSS use are the software supply chain vulnerabilities. A recent 2021 story by Alex Birsan[4] outlines a remarkably simple software supply chain attack, which affected PayPal, Apple, Microsoft, and many other companies. In a nutshell, a developer analyzed several companies' commonly used software components that were developed internally by these companies. He then created public open source packages on GitHub with matching names but unintended functionalities. Surprisingly, many developers from these companies started using his public libraries instead, a confusion that can easily happen when open source software development is highly integrated with in-house development. This led to bugs and supply chain vulnerabilities across the board,

[1] TensorFlow is a free and open-source software library for machine learning maintained by Google and community—https://www.tensorflow.org/.

[2] React is an open-source, front end, JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies—https://reactjs.org/.

[3] The GNU General Public License is a series of widely used free software licenses that guarantee end users the freedom to run, study, share, and modify the software—http://www.gnu.org/licenses/gpl-3.0.en.html.

[4] An article by Alex Birsan titled "Dependency Confusion: How I Hacked into Apple, Microsoft and Dozens of Other Companies"—https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610.

demonstrating that even successful industry players sometimes lack comprehensive open source governance.

What causes the above-mentioned risks and problems when using open source components in companies?

We aimed to answer this question in the first phase of our study. The answer essentially was the lack of open source software governance. Companies that did not have a structured and explicit approach for dealing with open source related questions were bound to have such problems. Among the companies we studied, some employees claimed that the use of open source software in their products was not allowed. However, even in such cases, developers used open source libraries and components simply without authorization or any checks, thus exacerbating the problem.

We then set out to find the recommendations for comprehensive open source governance. On the one hand, while many researchers recognized the issue [12–14], the academic literature with actual FLOSS governance guidelines were scarce and often limited to the "hot" topics, such as licensing [3, 15]. On the other hand, practitioners were filling the gap with their own experience reports [10, 16], which, however, did not follow rigorous or scientific methods for the elicitation of the recommended practices.

Seeing the industry relevance and need for this research, as well as acknowledging its potential business impact, we set out to study open source governance at companies with an advanced understanding of the topic. Based on the state-of-the-art literature and our preliminary industry interviews, we identified four key subtopics: getting started with open source governance, inbound governance, supply chain management, and outbound governance. Accordingly, we asked the following research questions:

- RQ1: How should companies using open source components in their products get started with open source governance based on existent industry best practice?
- RQ2: How should companies using open source components in their products govern the inbound aspects of the FLOSS use?
- RQ3: How should companies using open source components in their products govern their software supply chains?
- RQ4: How should companies using open source components in their products govern the outbound aspects of the FLOSS use?

Our goal was to gather expert knowledge from the industry through interviews, observations, and primary materials analysis, in order to distill and propose a set of industry best practices for open source software governance.

This phase of our study took about two years and resulted in more than 80 recommended industry best practices for different aspects of FLOSS governance, including:

- Transition toward open source governance
- IP-at-risk analysis
- Component search

- Component approval
- Component reuse
- Bill of materials management
- License compliance and more

Following the qualitative survey method by Jansen [17], we collected and analyzed data from more than 20 expert interviews at 15 companies, primary materials, and observation notes. This resulted in a proposed theory of industry best practices for corporate open source governance.

From an academic view, we built a theory, but we wanted to go beyond that and see our findings actually used by companies. To do this, we cast our theory as a practical handbook for open source governance that can be easily customized and used by practitioners in their companies. In each subsection of this chapter, we present best practice subsets from the handbook.

We concluded our study by conducting three evaluation case studies at large, multinational companies based in Germany. We followed the multiple-case case study method by Yin [18]. The collaboration with the selected companies was essential for the successful evaluation of our proposed best practices. We worked for over two years with the case study companies to implement parts of our open source governance handbook at different production projects. Over time, we observed how these companies with little to no previous understanding of FLOSS governance were able to establish internal processes and tools to efficiently and comprehensively manage their open source use. In the course of the implementation, we identified the lacking aspects of our theory and addressed them. At the same time, we helped these companies take their first steps toward FLOSS governance, which increased their benefits from using open source components while limiting the potential risks. In the following subsections, we couple the proposed best practices with their implementation reports at the case study companies.

Observing the actual industry impact of our research work was very rewarding, and we are happy to share our experience in this chapter, which is structured as follows. Section 2 focuses on the discovery and capture of industry best practices for open source governance including subsections on getting started with FLOSS governance, supply chain management. Section 3 goes on to describe the case studies where the proposed best practices were applied. Section 4 concludes the paper outlining the potential directions for future research.

2 Distilling Industry Best Practices

In the first phase of our study we developed a theory of industry best practices for corporate open source governance based on expert interviews, primary materials, and observations we conducted at 15 companies worldwide. These companies all had an advanced understanding of open source governance, included a form of open source program office responsible for establishing and executing open source

governance processes, and were willing to be interviewed for our study. We selected these companies from a larger sample of 140 companies in our professional network. We first classified these companies using different predefined criteria such as the size of the company, product types, business models, and more. We aimed for a polar theoretical sampling that would result in companies with different profiles and from different industries. As a result we had a mix of small, medium, and large companies from varying domains such as automotive, consulting, and enterprise software. The details of the sampling and method description can be found in the dissertation [19].

After analyzing the collected data following Jansen's qualitative survey method [17] and using qualitative data analysis tools such as MAXQDA,[5] we distilled the frequent best practices highlighted by experts. Figure 1 illustrates the hierarchical structure of the distilled best practice categories and subcategories.

Answering the research questions RQ1-RQ4 we addressed the main concepts of open source governance proposing subsets of industry best practices for each category. As an example, we can consider the Supply Chain Management (SCM) category in Fig. 1. Answering RQ3, we analyzed how expert companies established their SCM policy and processes focused on presenting governance and corrective governance. The former would include among other things recommendations for supplier selection with open source governance aspects in mind. The latter would focus on assessing the risks of supplier vulnerabilities and mitigating potential risks. Each subcategory (leaf in the tree) would then include individual best practices presented in the following subsections. An example of a best practice covering the choice of the right supplier as a preventive SCM measure is presented in Table 1.

Together all the best practices we identified form a handbook that can be modified and applied by any company that has software as part of their products. Additionally, to make the handbook more applicable for practitioners, we constructed workflows of interconnected best practices. Customizing and then following such a workflow would make the application of the handbook at a company easier. A sample page from the handbook is presented in Fig. 2.

2.1 Getting Started with FLOSS Governance

Answering the research question RQ1, we addressed a common problem companies have when faced with the issue of open source software governance: where does one start. The first part of our handbook is devoted to getting started with corporate open source governance.

In this "Getting Started" category, we identified the following best practices:

- Product Analysis (OSGOV-GETSTA-PROANA)—eight best practices
- IP-at-Risk Analysis (OSGOV-GETSTA-IPRISK)—nine best practices

[5] MAXQDA is a software program designed for computer-assisted qualitative and mixed methods data, text, and multimedia analysis—https://www.maxqda.com/.

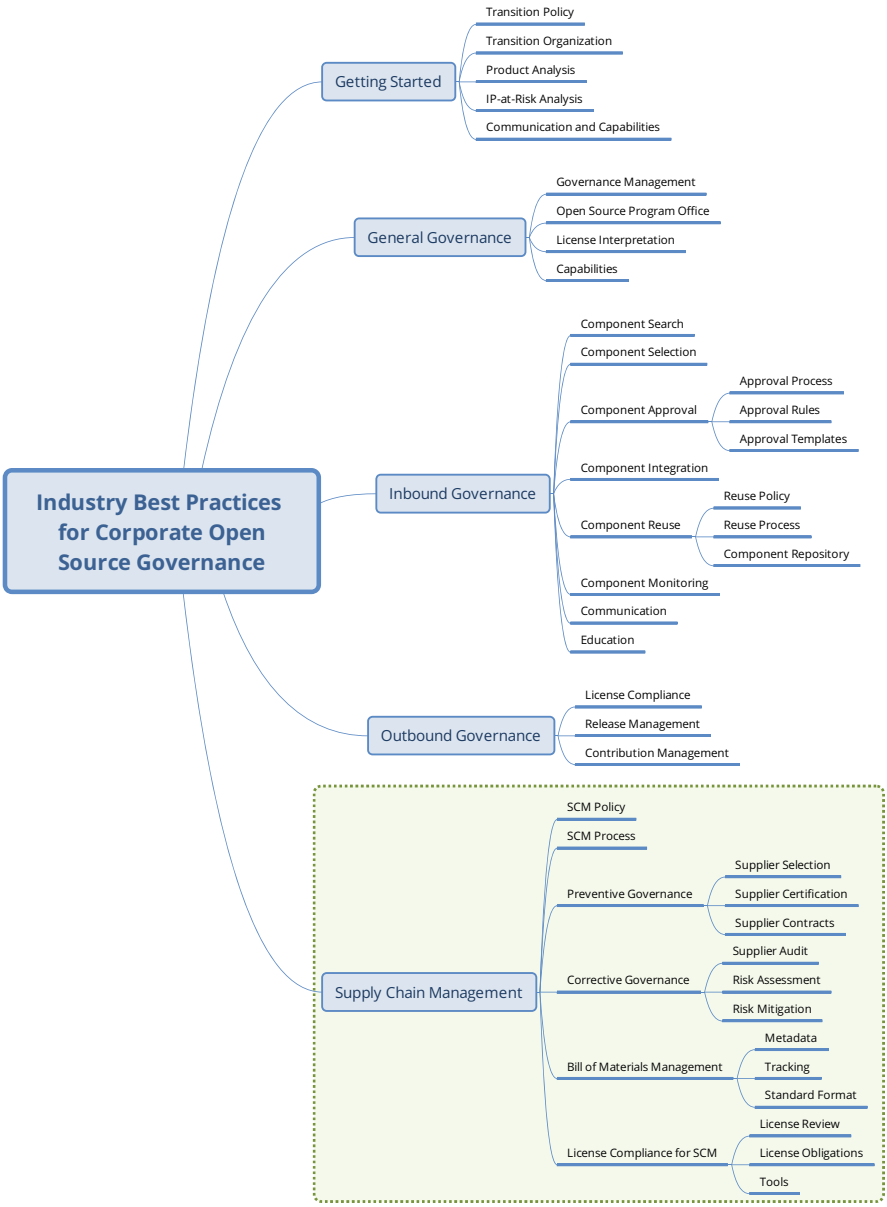

Fig. 1 Distilling industry best practices: Key concepts

- Transition Policy (OSGOV-GETSTA-TRAPOL)—three best practices
- Transition Organization (OSGOV-GETSTA-TRAORG)—eight best practices
- Communication and Capabilities (OSGOV-GETSTA-COMCAP)—five best practices

Table 1 OSGOV-SUCHMA-PREGOV-1. Choose the right supplier

Name	Choose the right supplier
Actor	Supply chain management responsibles, IT department, procurement department
Context	Virtually all companies use supplied software components as part of their products. Not all suppliers are the same in terms of open source governance and compliance. Choosing a supplier without open source governance consideration can result in functionally superior software with open source components that are not compliant with the company's license use case pairs
Problem	If supplied code causes open source governance and compliance risks, you will have to either change your supplier or address the risks in cooperation with the supplier after the delivery. How can such situations be prevented?
Solution	To prevent potential issues with FLOSS governance and compliance you should choose the right suppliers that are aware and mature in terms of governance and compliance, as well as experienced in using open source components in their products. To do this, you need to → *assess open source governance and compliance awareness and maturity* which can be done by → requesting supplier certification or self-certification from potential suppliers. To establish a consistent approach for preventive governance → *design supplier contracts with open source governance aspects in mind* and use the governance related clauses in case of license non-compliance by a supplier. The latter can be used for corrective governance, namely to → *trigger supplier contract clauses and get the supplier to take care of the issue*

The Product Analysis subcategory included practices for an initial scanning of the open source components already in use at a company. Even if the company had a policy discouraging the use of open source software, more often than not, there are a considerable number of such components that were previously undetected or undocumented. This cannot be ignored when getting started with FLOSS governance. You need to take stock of the overall FLOSS use before establishing the governance processes.

Once the initial assessment is conducted, companies need to perform an IP-at-Risk Analysis, which covers the assessment of potential licensing issues, copyright violations, or other inconsistencies. The identified risks need to be mitigated at this stage either by complying with the component license or by removing or encapsulating the component.

Afterwards, the company can start the transition from the unstructured use of open source software to basic governance defined in a Transition Policy and operationalized through the proposed Transition Organization best practices. During the whole transition, companies need to pay close attention to communicating the changes to all the stakeholders and building internal capabilities where needed.

An example best practice from the Getting Started category is presented in Table 2, which covers the basic process that the project architect will use for reporting and assessing ongoing additions of open source components during the transition.

A. Process Template for Supplier Management 1

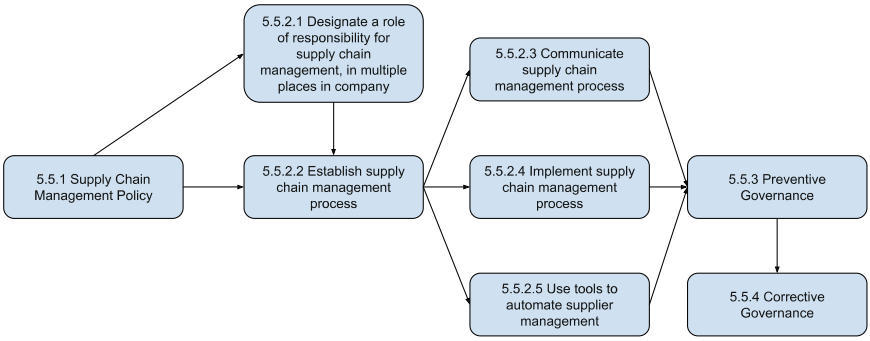

Figure 10. Supplier Management Process

B. Process Template for Supplier Management 2

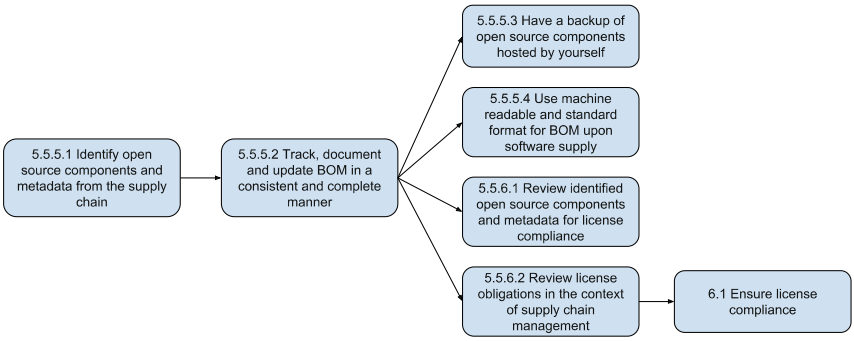

Figure 11. Supplier Management Process

Fig. 2 Distilling industry best practices: Key concepts

All the proposed best practices can be traced to the primary data we analyzed in the first stage of our study. Here is an example of such a trace from Company 14s[6] legal counsel responsible for open source compliance talking about the specifics of establishing a process of continuous reporting and assessment of open source components:

[6] Company and interviewee names were anonymized per their request.

Table 2 OSGOV-PROANA-1.2. Establish a process of continuous reporting and assessment

Name	Establish a process of continuous reporting and assessment
Actor	Transition manager and/or project architect
Context	You already → *used one mandatory survey for initial assessment*. Now you need a process for continuous reporting and assessment of any open source usage during the transition
Problem	The transition needs to prepare the company for fully structured FLOSS governance. However, during the transition how should the process of continuous reporting and assessment look like?
Solution	Establish a process of continuous reporting and assessment that involves defined and easy-to-follow steps for developers when using new open source components during the transition. This can be achieved using a product architecture model (a meta-model for all governance related information such as license information, copyright noticed, export restrictions, etc.), bill of materials documentation, questionnaires or forms, etc. The process should help: • Continuously report new use of open source components during transition • Automate this reporting as much as possible, by → *selecting and using governance tools for automation* • Continuously assess new use of open source components during transition – Assess license compliance – Assess copyright notices – Assess export restrictions – Assess software supply chains • Document the assessment findings • Share the reported use of open source and documented assessment findings

> When our developers are reporting the open source via [our internal tool], there is always the main file which is also mentioned in the license file which is also computed by GitHub or by the community behind. And with this scan tooling, we cross-check the whole software, so we definitely see, okay, that's not only the MIT license which is mentioned in the license file but also other licenses, so the GPL files. And, then, we're talking to our developer which is reporting the open source. In most cases, the developer says, no to the GPL files, we don't use it, we only use the MIT file. And, so, they need to cross check what files they use, and what licenses are used by them. – Company 14

More best practices on getting started with open source governance can be found in the dissertation [19] or in our previous paper on the subject [20]. Furthermore, we addressed the research questions RQ2 and RQ3 on inbound and outbound governance in our previous papers [21] and [22], as well as in the dissertation [19].

2.2 Supply Chain Management

Answering the research question RQ2, we focused on the core topic of our research—software supply chain management in the scope of FLOSS governance.

Software supply chains include both proprietary code (developed in-house or supplied) and open source code. Modern software supply chains have multiple tiers where each supplier has multiple suppliers on the deeper tiers. In each instance, open source components are used and integrated into the larger software package that ends up in the software of the end user sold by the OEM (Original Equipment Manufacturer) company. In a nutshell, in case of issues due to poor FLOSS governance down the supplier hierarchy, the whole responsibility can rarely be put on suppliers.

We can consider a car example. The car manufacturer has commissioned a supplier to develop an infotainment system for the car including navigation, music, and other functionalities. The supplier (or their suppliers) uses open source libraries for the graphical interface and a Linux-based operating system, both of which are licensed under the GPL v3 (GNU General Public License version 3) license. This license requires any user of the code to publish it under the same license and to disclose the original copyright holders. If the suppliers fail to mention this to the automotive company and the company does not do its own due diligence in terms of FLOSS governance, a copyright holder of the original software under the GPL license can sue the company for license non-compliance. As a result, the company would face steep legal or operational costs as they would either have to settle or update their software to comply with the license.

In this category we distilled industry best practices for the following subcategories:

- Supply Chain Management Policy (OSGOV-SUCHMA-SCMPOL)—three best practices
- Supply Chain Management Process (OSGOV-SUCHMA-SCMPRO)—five best practices
- Preventive Governance (OSGOV-SUCHMA-PREGOV)—four best practices
- Corrective Governance (OSGOV-SUCHMA-CORGOV)—four best practices
- Bill of Materials Management (OSGOV-SUCHMA-BOMMAN)—four best practices
- License Compliance for Supply Chain (OSGOV-SUCHMA-LICCOM)—two best practices

To go one level deeper, we can consider the specific best practices focused on corrective governance:

- OSGOV-SUCHMA-CORGOV-1. Audit your supply chain
- OSGOV-SUCHMA-CORGOV-2. Mitigate identified risks
- OSGOV-SUCHMA-CORGOV-2.1. Assess risks in accordance to the supply chain management policy
- OSGOV-SUCHMA-CORGOV-2.2. Trigger supplier contract clauses and get the supplier to take care of the issue
- OSGOV-SUCHMA-CORGOV-2.3. Do not run your supplier out of business

An example best practice in this subcategory is presented in Table 3. While somewhat counter-intuitive, we found an industry best practice discouraging an

extreme pressure on the suppliers in case of FLOSS governance issues identified too late. If you were to put the whole responsibility on the supplier, you are running a chance of bankrupting the supplier, which could leave your company in a worse situation. As a result, you still have to deal with the litigation but also need to find a different supplier or maintain their code on your own.

Table 3 OSGOV-SUCHMA-CORGOV-2.3. Do not run your supplier out of business

Name	Do not run your supplier out of business
Actor	OSPO (Open Source Program Office), Lawyer/legal counsel
Context	You have identified compliance and governance risks in your supply chain and → *assessed these risks in accordance with the supply chain management policy*. For certain critical risks you → *triggered supplier contract clauses to take care of the issue*
Problem	What actions should you not take when addressing the identified risks of non-compliance by a supplier?
Solution	Most companies have suppliers that are smaller than themselves, thus giving them higher negotiation power over the suppliers. This means that in case of non-compliance with open source licenses in the supplied code, you can easily force your supplier to fix the risk causing software non-compliance. You can even sue your supplier and get compensation. However, you should be careful not to endanger the operation of the supplier company
	If you run your supplier out of business by pressuring them with lawsuits or financial pressure, you can end up with a binary instead of a source code and no ability to maintain or update the software that was causing the non-compliance issue in the first place. Most software is not supplied as source code, but rather as a binary in order to protect the intellectual property of the supplier that makes money by selling a different version of the product that uses its know-how in the form of source code. If a company goes bankrupt, you might have to look for another supplier, which is costly and time consuming. In a nutshell, do not run your supplier out of business, when possible. Alternatively, make sure to get the source code in case of the supplier bankruptcy or before changing the supplier to avoid the above mentioned risk

More practices on the topic of supply chain management can be found in the dissertation [19] or in our previous publication [23].

3 Applying Industry Best Practices

In the second phase of our study we applied our handbook for open source governance at three companies in order to evaluate our findings in a real-life context using the case study methodology by Yin [18]. We identified three Germany-based companies that had little to no experience of open source governance, but had products incorporating software. These companies were willing to learn and implement FLOSS governance processes following our recommendations. On the one hand, this was an opportunity for them to learn from the leading companies in

terms of open source use and governance. On the other hand, it was an opportunity for us to have an actual industry impact by having our theory implemented and testing it in real life. The three companies were at different stages in terms of FLOSS governance. Company A had no prior experience of structured open source governance, while Company B and C had some basic experience. Company B, in particular, was dealing with many software suppliers and needed support in that domain.

Further details on the case study company profiles and specifics can be found in the dissertation [19].

3.1 Case Study A

Once the Getting Started part of the handbook was completed, we started the evaluation phase that was running in parallel to the further theory building and handbook extension. We implemented a subset of the proposed Getting Started best practices at Company A, namely at five of their divisions with a focus on Division A.1 (with aerospace products including software systems and components).

In the course of the transition toward open source governance at Division A.1, a newly established open source program office (OSPO)[7] followed our Getting Started best practice *OSGOV-GETSTA-PROANA-3.1. Run open source use analysis in products*. An R&D manager who was part of the OSPO ran an initial analysis of the current FLOSS use at the division using an open source tool called FOSSology[8] [24]. An excerpt from this analysis is presented in Fig. 3. Following the best practice was well worth it, as the OSPO identified a number of problematic components that were previously overlooked or ignored, including but not limited to open source software licensed under GPL and AGPL[9] (GNU Affero General Public License) licenses. These components were later reviewed and analyzed individually.

Case Study A was extensive spanning over 2.5 years and enabled us to run a longitudinal study into the implementation of various parts of our handbook across five different divisions operating in different industries and in different countries including Germany, Mexico, and China. Figure 3 presents only a brief snippet of an evaluation artifact. More artifacts and thorough details can be found in the dissertation [19], including a complete open source governance process that Company A created when customizing our handbook.

[7] Open Source Program Office (OSPO) is a typical organization unit responsible for managing the internal and external aspects of corporate open source governance, often composed of software developers, software architects, lawyers, and product managers.

[8] FOSSology is an open source license compliance software system and toolkit that can be used to scan software for license, copyright, and export control issues—https://www.fossology.org/.

[9] GNU Affero General Public License is based on the GPL license but has more restrictive license terms—https://www.gnu.org/licenses/agpl-3.0.en.html.

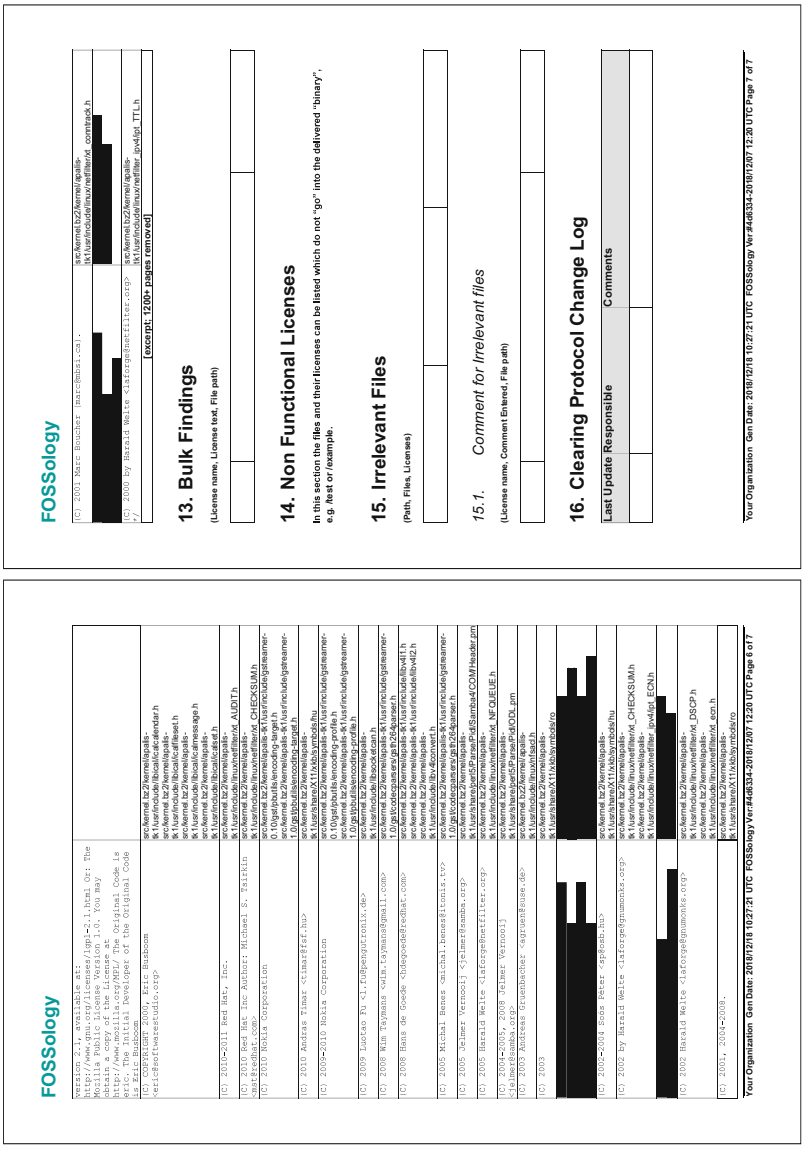

Fig. 3 Applying industry best practices: Initial open source use assessment at Case Study A

3.2 Case Study B

Company B was an enterprise software company operating internationally but based in Germany. With nearly 5000 employees, the company had different approaches for open source governance across different teams and business units. At the

time of our engagement, there was an initiative of establishing a centralized organizational unit that would deal with everything related to open source software. This team has been acquiring new FLOSS governance responsibilities and was glad to collaborate with us in the scope of a research project. We introduced a subset of best practices for Supply Chain Management from our handbook and observed their implementation at the company. Similar to Company A, we did not conduct the actual implementation, but rather observed and supported it. This enabled a more unbiased evaluation of how our handbook would perform in a real-life setting of a company.

Company B followed a number of our best practices establishing internal processes for supplier evaluation with FLOSS governance in mind, as well as optional supplier certifications, bill-of-materials (BOM) management, and more. As a notable example, the software supplier manager at Company B followed our best practice *OSGOV-SUCHMA-PREGOV-1.1. Assess open source governance and compliance awareness and maturity* to create a supplier questionnaire that included the key aspects recommended by our handbook. While abstract, the proposed best practices served as a good basis to develop new supplier guidelines that were sent to actual suppliers. Figure 4 presents an excerpt of the supplier questionnaire (its first page with the table of contents). The questionnaire included the following points recommended by our handbook:

- License information (including open source licenses)
- Level of open source awareness
- Technologies used in the development process
- Integration of third party software into supplier code
- Bill-of-materials information

Our best practice did not cover all the aspects required by Company B, so they added points on quality management (including information on ISO certifications), and sales process.

To see more details of our evaluation case studies, as well as handbook implementation artifacts, see the dissertation [19].

Licensor Name	
Name of product	
Version of product	
Is the product white labeled	[_] yes [_] no

Fig. 4 Applying industry best practices: Supplier questionnaire at Case Study B

4 Conclusion

This section concludes the book chapter with the results' highlights from our work on open source software governance in companies. In the first stage of the study, we distilled some best practices used in the industry for open source governance. We captured and presented them as part of a larger handbook for FLOSS governance that focused on getting started with governance, managing the inbound and outbound aspects, as well as supply chain management. The handbook was the practical artifact we developed to make our research results more relevant and applicable for practitioners, who could customize and use our best practices. The handbook consisted of topical categories and subcategories. Individual best practices were arranged into workflows that would enable an easier execution at a company.

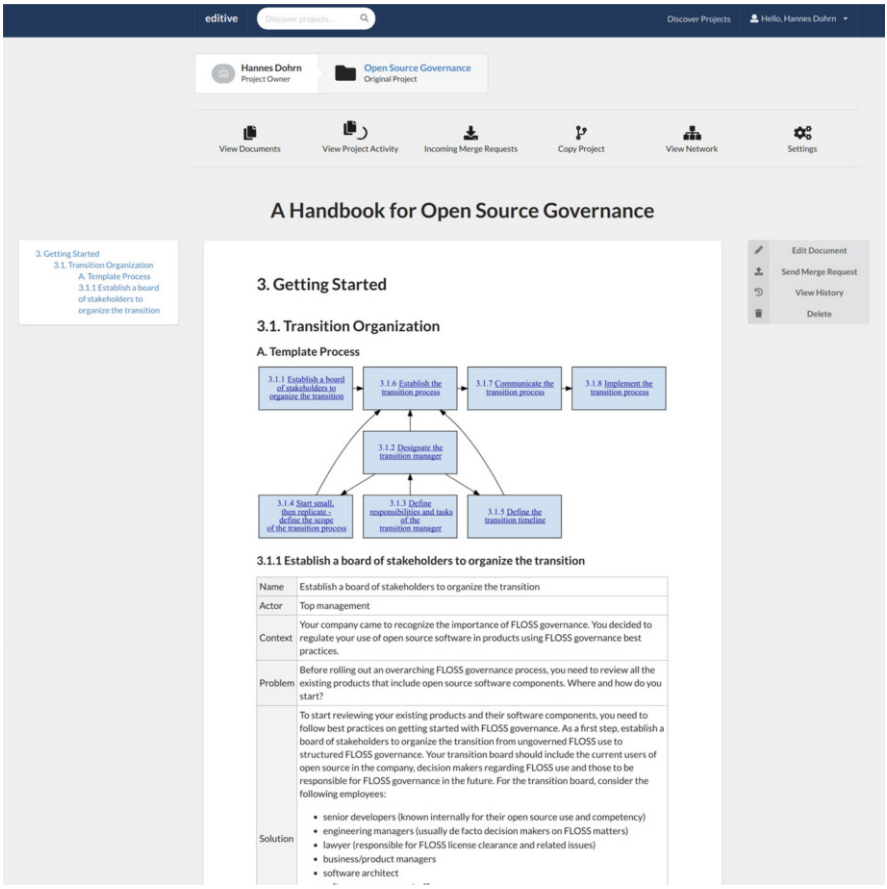

Fig. 5 Tooling for handbook representation and forking: Editive—https://editive.com/en/

In the second stage of the study, we took parts of the handbook to companies in Germany willing to put our findings to test in their production projects. We conducted three case studies, where different best practice subsets were implemented and evaluated in a real-life setting. The case studies demonstrated the pitfalls of some best practices that we later addressed. They also demonstrated the limitations of best practice transferability across companies. As a result, our handbook and its best practices had to be abstract in nature and not specific for one industry domain. Instead, we aimed for broad applicability and customization.

Going beyond the traditional academic work of theory building and working closely with the industry to evaluate our findings, we also faced some setbacks. Namely, Case Study C partially failed due to the misaligned expectations from the project, as well as different visions for the open source governance handbook. We learned that large shifts in company software architectures or processes are no easy feat. They need to be thoroughly prepared and be backed by different stakeholders across the company. And even in such cases, success is not a given. However, we were able to learn from both our successes and failures during the three years of working on this topic.

We see future opportunities for extending the research into other aspects of both open source governance and related topics, such as corporate open sourcing or license compliance. We also want to highlight the potential of the handbook method for other studies in software engineering and computer science. Finally, we see potential tooling support for the handbook that would make it even easier for companies to use and modify the best practices across their organizational complexities. In fact, we collaborated with a startup (Editive—https://editive.com/en/) based on a spin-off project at our university to create a *forkable* prototype of our handbook to present to our industry partners. A screenshot is presented in Fig. 5. The further development of such tools can make introducing and maintaining open source governance easier and more streamlined for any company in the future.

Acknowledgments This was not an individual effort. Throughout the whole research, many people supported me—my family, my friends, my colleagues, and industry partners. I want to especially thank my professor Dirk Riehle and my colleagues Ann Barcomb, Andreas Bauer, Fariba Bensing, Maximilian Capraro, Hannes Dohrn, Michael Dorner, Andreas Kaufmann, Daniel Knogl, and Georg Schwarz for their contributions to this research.

References

1. Ruffin, C., Ebert, C.: Using open source software in product development: a primer. IEEE Softw. **21**(1), 82–86 (2004)
2. Lin, L.C.-H., Shen, N.: Copyleft referring to GPL-3.0 was cited as a defense method in Chinese intellectual property court in Beijing. Int. Free Open Source Softw. Law Rev. **10**(1), 1–7, (2019)
3. 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, Silver Spring (2009)

4. Merilinna, J., Matinlassi, M.: Assessing the role of open source software in the European secondary software sector: a voice from industry. In: 1st International Conference on Open Source Systems (2005)
5. Chen, W., Li, J., Ma, J., Conradi, R., Ji, J., Liu, C.: An empirical study on software development with open source components in the Chinese software industry. Softw. Process Improv. Practice **13**(1), 89–100 (2008)
6. Agerfalk, P.J., Deverell, A., Fitzgerald, B., Morgan, L.: State of the art and practice of open source component integration. In: 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06), pp. 170–177. IEEE, Piscataway (2006)
7. Akkanen, J., Demeter, H., Eppel, T., Iványi, Z., Nurminen, J.K., Stenman, P.: Reusing an open source application—practical experiences with a mobile CRM pilot. In: IFIP International Conference on Open Source Systems, pp. 217–222. Springer, Berlin (2007)
8. Ayala, C., Hauge, Ø., Conradi, R., Franch, X., Li, J., Velle, K.S.: Challenges of the open source component marketplace in the industry. In: IFIP International Conference on Open Source Systems, pp. 213–224. Springer, Berlin (2009)
9. Stol, K.-J., Ali Babar, M.: Challenges in using open source software in product development: a review of the literature. In: Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, pp. 17–22. ACM, New York (2010)
10. Popp, K.M.: Best Practices for commercial use of open source software: business models, processes and tools for managing open source software. BoD–Books on Demand (2015)
11. Helmreich, M.: Best practices of adopting open source software in closed source software products (2011)
12. Kemp, R.: Towards free/libre open source software governance in the organization. IFOSS L. Rev. **1** (2009)
13. Markus, M.L.: The governance of free/open source software projects: monolithic, multidimensional, or configurational? J. Manag. Governance **11**(2), 151–163 (2007)
14. Gangadharan, G., D'Andrea, V., De Paoli, S., Weiss, M.: Managing license compliance in free and open source software development. Inform. Syst. Front. **14**(2), 143–154 (2012)
15. Alspaugh, T.A., Asuncion, H.U., Scacchi, W.: Analyzing software licenses in open architecture software systems. In: Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, pp. 54–57. IEEE, Piscataway (2009)
16. Peters, S.: Best practices for creating an open source policy (2010)
17. Jansen, H.: The logic of qualitative survey research and its position in the field of social research methods. Forum Qualitative Sozialforschung/Forum: Qualitative Social Research **11**(2), (2010)
18. Yin, R.K.: Case Study Research and Applications: Design and Methods. Sage Publications, New York (2017)
19. Harutyunyan, N.: Corporate Open Source Governance of Software Supply Chains. doctoralthesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) (2019)
20. Harutyunyan, N., Riehle, D.: Getting started with floss governance and compliance: a theory of industry best practices. In: Proceedings of the 15th International Symposium on Open Collaboration, Forthcoming, 2019
21. Harutyunyan, N., Riehle, D.: Industry best practices for FLOSS governance and component reuse. In: Proceedings of the 24th European Conference on Pattern Languages of Programs. ACM, New York (2019)
22. Harutyunyan, N., Riehle, D.: Industry best practices for component approval in floss governance. In: Proceedings of the 25th European Conference on Pattern Languages of Programs. ACM, New York (2020)
23. Harutyunyan, N.: Managing your open source supply chain-why and how? Computer **53**, 77–81 (2020)
24. Gobeille, R.: The FOSSology project. In: Proceedings of the International Working Conference on Mining Software Repositories, pp. 47–50. ACM, New York (2008)