Corporate Open Source Governance of Software Supply Chains

Open Source Governance der Software-Lieferketten in Firmen

Der Technischen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg
zur
Erlangung des Doktorgrades

Doktor-Ingenieur (Dr.-Ing.)

vorgelegt von

Nikolay Harutyunyan

Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung:	30. September 2019
Vorsitzender des Promotionsorgans:	Prof. Dr.-Ing. Reinhard Lerch
Gutachter:	Prof. Dr. Dirk Riehle
	Prof. Dr. Björn Lundell

Abstract

Today most software products incorporate free/libre and open source software (FLOSS) components. FLOSS components are used in development infrastructures, generic and non-differentiating features and functionalities, web browsers, databases, and operating systems. Some software products are built on top of FLOSS frameworks. Open source software usage often ensures lower cost, higher quality, and quick availability, especially when using generic software components and libraries.

Using open source software in products comes with legal, business, and technical risks, however. A major challenge for companies is understanding and complying with licenses and regulations related to the FLOSS components they use in their products. On a deeper level, these issues go beyond the in-house software development, including complete software supply chains and FLOSS components used within them. The potential risks of using open source components can result in litigation due to open source license non-compliance, copyright infringement, or loss of intellectual property. There are several ongoing court cases in Germany, in China, and in the USA, which highlight the significance of the above-mentioned risks.

Companies can address the risks of FLOSS use by establishing *corporate open source governance* – a set of processes, best practices, and tools employed by companies to govern the use of FLOSS components as parts of their commercial products while minimizing their risks and maximizing their benefit from such use. Corporate open source governance covers topics of open source component selection and approval, ensuring open source license compliance through code scans and audits, as well as bill of materials management focused on open source component and their metadata, etc.

The goal of our research project was to build a methodological framework for corporate open source governance in companies with software-intensive products. In the scope of this dissertation, we researched the state of the art regarding the corporate open source governance in the literature, built a theory of industry best practices for corporate open source governance based on expert interviews and other primary materials, and evaluated the proposed theory through a multiple-case case study at German companies operating internationally.

In the first stage, we studied the state of the art in academic research through a literature review of 87 publications on the topic. We conducted a qualitative data analysis of the papers, deriving the core concepts of corporate open source governance from the literature – risks of the ungoverned FLOSS use, getting started with FLOSS governance, inbound governance, supply chain management, outbound governance, and general governance. We then mapped the reviewed papers to the

identified core concepts and presented highlights on each concept, which were later com-pared with the insights from the proposed theory.

In the second stage, we asked the overarching research question of how companies using open source components in their products should govern this use based on the industry best practices. Taking a practice-based approach, we conducted a qualitative survey that included 20 primary materials (published governance guidelines, white papers, slides) and 21 expert interviews at 15 companies with an advanced understanding of open source governance. Based on the findings from the qualitative survey, we built a theory of industry best practices for corporate open source governance, with a particular focus on supply chain management. Our theory proposes industry best practices on the core topics of FLOSS governance in companies – getting started with corporate open source governance, inbound governance, outbound governance, general governance, and the focal topic of the study – supply chain management in the context of open source governance. Going beyond the textual presentation of the theory, we also presented our findings in an actionable and industry-friendly format of interconnected best practice patterns that formed a handbook for corporate open source governance. We attached excerpts from our governance handbook in the appendix of this dissertation.

In the third stage, we evaluated the proposed theory through a multiple-case case study with a holistic design at two case studies in production-level projects at two large German companies that used open source software in their products, but lacked open source governance. Case Study A was a 2.5-year longitudinal study into a company that was just getting started with open source governance. This enabled us to evaluate the getting started and inbound governance aspects of our theory. The length of the study enabled us to thoroughly analyze the current use of open source software and its informal governance in the various divisions of the company, followed by the guided implementation of the industry best practices from our theory. We then observed how the best practices were applied in a real-life production environment. Case Study B was a one-year longitudinal study into a company that already has the fundamental framework for open source governance, but lacks processes and practices for governing the use of open source software from its supply chain. Using the initial situation assessment as a baseline, we implemented the industry best practices on supply chain management from our theory. We then observed the effectiveness of the proposed practices in improving FLOSS governance and the drawbacks of these practices. In both case studies, we evaluated our theory using the quality criteria of completeness, variability, structure, comprehension, understandability, applicability, relevance, significance, and usefulness.

We conclude the dissertation by discussing the key results highlighting the value of our contribution to both academia and industry. On one hand, our research publications enrich the academic research on open source governance. On the other hand, practitioners can follow the suggested best practices applying the governance handbook we developed. We also discussed limitations of this dissertation to both theory building and theory evaluation. We then suggested directions for further research that could enrich the topic of corporate open source governance, for example industry best practices for open source license compliance and its automation, setting up and managing an open source program office, open source component search and selection, and release management.

Zusammenfassung

Heutzutage enthalten die meisten Softwareprodukte Free/Libre- und Open-Source-Software (FLOSS)-Komponenten. FLOSS-Komponenten werden in Entwicklungsinfrastrukturen, generischen und nicht-differenzierenden Features und Funktionalitäten, Webbrowsern, Datenbanken und Betriebssystemen eingesetzt. Einige Softwareprodukte basieren auf FLOSS-Frameworks. Die Nutzung von Open-Source-Software gewährleistet oft niedrigere Kosten, höhere Qualität und schnelle Einsatzfähigkeit, insbesondere bei der Verwendung generischer Softwarekomponenten und Bibliotheken.

Der Einsatz von Open-Source-Software in Produkten birgt jedoch rechtliche, wirtschaftliche und technische Risiken. Ein großes Problem für Unternehmen ist das Verständnis und die Einhaltung von Lizenzen und Regularien im Zusammenhang mit den FLOSS Komponenten, die sie in ihren Produkten verwenden. Auf einer tieferen Ebene gehen diese Fragen über die eigene Softwareentwicklung hinaus, einschließlich kompletter Software-Lieferketten und der darin verwendeten FLOSS-Komponenten. Die potenziellen Risiken bei der Verwendung von Open-Source-Komponenten können zu Rechtsstreitigkeiten aufgrund von Verstößen gegen die Open-Source-Lizenz, Urheberrechtsverletzungen oder dem Verlust geistigen Eigentums führen. In Deutschland, China und den USA gibt es mehrere laufende Gerichtsverfahren, die die Bedeutung der oben genannten Risiken verdeutlichen.

Unternehmen können die Risiken der Nutzung von FLOSS adressieren, indem sie eine Reihe von Prozessen, Erfolgsmethoden und Tools einführen, um FLOSS-Komponenten als Teil ihrer kommerziellen Produkte zu verwenden, während sie ihre Risiken minimieren und ihren Nutzen aus einer solchen Nutzung maximieren. Corporate Open Source Governance umfasst Themen wie die Auswahl und Genehmigung von Open Source-Komponenten, die Sicherstellung der Einhaltung von Open Source-Lizenzen durch Code-Scans und Audits, die Stücklistenverwaltung mit Schwerpunkt auf Open Source-Komponenten und deren Metadaten, etc.

Ziel unseres Forschungsprojektes war es, einen methodischen Rahmen für Corporate Open Source Governance in Unternehmen mit softwareintensiven Produkten zu schaffen. Im Rahmen dieser Dissertation haben wir den Stand der Technik der Corporate Open Source Governance Literatur gründlich untersucht, eine Theorie der branchenspezifischen Erfolgsmethoden für Corporate Open Source Governance auf der Grundlage von Experteninterviews und anderen Primärmaterialien erstellt und die vorgeschlagene Theorie durch eine mehrteilige Fallstudie bei international tätigen deutschen Unternehmen bewertet.

In der ersten Phase haben wir den Stand der Technik in der Forschung durch eine Literaturrecherche von 87 Publikationen zu diesem Thema untersucht. Wir führten eine qualitative Datenanalyse der Papiere durch und leiteten die Kernkonzepte der Corporate Open Source Governance aus der Literatur ab – Risiken der unkontrollierten Nutzung von FLOSS, Einstieg in die FLOSS Governance, Inbound Governance, Lieferketten Management, Outbound Governance und General Governance. Wir haben dann die überprüften Papiere den identifizierten Kernkonzepten zugeordnet und Highlights zu jedem Konzept präsentiert, die wir mit den Erkenntnissen aus der vorgeschlagenen Theorie vergleichen.

In der zweiten Phase stellten wir die übergreifende Forschungsfrage, wie Unternehmen, die Open-Source-Komponenten in ihren Produkten verwenden, diese Nutzung auf der Grundlage der Erfolgsmethoden der Branche regeln sollten. Mit einem praxisorientierten Ansatz führten wir eine qualitative Studie durch, die 20 Primärmaterialien (veröffentlichte Governance-Richtlinien, Whitepaper, Folien) und 21 Experteninterviews bei 15 ausgewählten Unternehmen mit einem fortgeschrittenen Verständnis von Open Source Governance umfasste. Basierend auf den Ergebnissen der qualitativen Studie haben wir eine Theorie der branchenspezifischen Erfolgsmethoden für Corporate Open Source Governance mit besonderem Fokus auf Lieferketten Management entwickelt. Unsere Theorie schlug branchenspezifische Erfolgsmethoden zu den Kernthemen von FLOSS Governance in Unternehmen vor - den Einstieg in Corporate Open Source Governance, Inbound Governance, Outbound Governance, General Governance und das Schwerpunktthema der Studie - Lieferketten Management im Kontext von Open Source Governance. Über die textuelle Darstellung der Theorie hinaus präsentierten wir unsere Ergebnisse auch in einem umsetzbaren und industriefreundlichen Format von vernetzten Erfolgsmethoden-Mustern, die ein Handbuch für Corporate Open Source Governance bildeten. Auszüge aus unserem Governance-Handbuch haben wir im Anhang dieser Dissertation beigefügt.

In der dritten Phase bewerteten wir die vorgeschlagene Theorie durch eine mehrteilige Fallstudie mit einem ganzheitlichen Design in zwei Fallstudien in Projekten auf Produktionsniveau bei zwei großen deutschen Unternehmen, die Open-Source-Software in ihren Produkten verwendeten, aber keine Open-Source-Governance hatten. Fallstudie A war eine 2,5-jährige Längsschnittstudie über ein Unternehmen, das gerade erst mit Open Source Governance begann. Dies ermöglichte es uns, die Aspekte des Getting Started und der Inbound-Governance unserer Theorie zu bewerten. Die Länge der Studie ermöglichte es uns, den aktuellen Einsatz von Open-Source-Software und deren informelle Governance in den verschiedenen Bereichen des Unternehmens gründlich zu analysieren, gefolgt von der gezielten Umsetzung der branchenspezifischen Erfolgsmethoden aus unserer Theorie. Anschließend beobachteten wir, wie die Erfolgsmethoden in einer realen Produktionsumgebung angewendet wurden. Fallstudie B war eine einjährige Längsschnittstudie über ein Unternehmen, das bereits über den grundlegenden Rahmen für Open Source Governance verfügt, aber keine Prozesse und Praktiken für die Regelung der Nutzung von Open Source Software aus seiner Lieferkette hat. Ausgehend von der Ausgangssituation als Grundlage haben wir die branchenspezifischen Erfolgsmethoden für das Lieferketten Management aus unserer Theorie implementiert. Anschließend beobachteten wir die Wirksamkeit der vorgeschlagenen Praktiken zur Verbesserung der FLOSS Governance und die Nachteile dieser Praktiken. In beiden Fallstudien haben wir unsere Theorie anhand der Qualitätskriterien Vollständigkeit, Variabilität, Struktur, Verständnis, Verständlichkeit, Anwendbarkeit, Relevanz, Bedeutung, und Nützlichkeit bewertet.

Abschließend diskutieren wir die wichtigsten Ergebnisse, die den Wert unseres Beitrags für Wissenschaft und Industrie verdeutlichen. Zum einen bereichern unsere Forschungspublikationen die wissenschaftliche Forschung zur Open Source Governance. Auf der anderen Seite können Praktiker den vorgeschlagenen Erfolgsmethoden folgen und das von uns entwickelte Gover-nance-Handbuch anwenden. Wir diskutierten auch die Grenzen dieser Dissertation sowohl für den Theorieaufbau als auch für die Theoriebewertung. Dann schlugen wir Richtungen für weitere Forschungen vor, die das Thema Corporate Open Source Governance bereichern könnten, z.B. branchenspezifische Erfolgsmethoden für Open Source Lizenzkonformität und deren Automatisierung, Einrichtung und Verwaltung eines Open Source Programmbüros, Open Source Komponentensuche und -auswahl sowie Release-Management.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AGPL	GNU Affero General Public License
B2B	business-to-business
B2C	business-to-consumer
BoM	bill of materials
BP	(proposed) best practice
BRR	Business Readiness Review
BSD	Berkeley Software Distribution (license)
BU	business unit
FAU	Friedrich-Alexander-University Erlangen-Nuremberg
FLOSS	free/libre open source software
GCC	GNU Compiler Collection
GENGOV	general governance
GETSTA	getting started (with corporate open source governance)
GPL	GNU General Public License
GT	governance tool providers
ICSE	International Conference on Software Engineering
IDE	integrated development environment
IFOSSLR	International Free and Open Source Software Law Review
INBGOV	inbound governance
IoT	internet of things
IP	intellectual property
IT	information technology (department)
LGPL	GNU Lesser General Public License
MC	management consulting
MIS	Management Information Systems

OEM	original equipment manufacturer
OP	other products incorporating software
OpenGL	Open Graphics Library
OS	open source
OSADL	Open Source Automation Development Lab eG
OSCT	Open Source Compliance Team
OSF	open source foundation
OSGOV	(corporate) open source governance
OSPO	Open Source Program Office
OSS	open source software
OSSCO	Open Source Software Compliance Officer
OUTGOV	outbound governance
PROANA	product analysis
PT	process template
QDA	qualitative data analysis
RQ-TE	research question – theory evaluation
RQ	research question
SaaS	software as a service
SCM	supply chain management
SDS	software development service
SP-CS	software product vendor for closed source software
SP-OS	software product vendor for open source software
SPDX	Software Package Data Exchange
SUCHMA	supply chain management
SWOT	strengths, weaknesses, opportunities, threats
TPP	third-party program
UI	user interface
WP	work package

Dedicated to Nikolay Smbatovich, Armen, Hasmik, Viktor and Shushan

Acknowledgments

I AM GRATEFUL to my academic mentors, colleagues, and industry partners for the support, academic contributions, and fun memories in the course of my doctoral studies.

I thank my professor Dirk Riehle for the invaluable guidance from the very beginning of my project and for the freedom to work on the dissertation, as well as for putting together a research group that was a pleasure to work with.

I thank my colleagues at the Open Source Research Group – Ann Barcomb, Andreas Bauer, Fariba Bensing, Maximilian Capraro, Hannes Dohrn, Michael Dorner, Andreas Kaufmann, Daniel Knogl, and Georg Schwarz. Thank you *each* for the support, advice, feedback, constructive criticism, interesting conversations, and lifelong memories with and without 30-year-old port. Ann and Andi, thank you for the research collaboration and your contributions to my thesis. Andreas, thank you for being a great office mate and cheerleader. Max, Hannes, Michi, thank you for the numerous discussions on research methods and ideas. I also thank Annika Donhauser for her diligence and contribution to my research project as a working student.

I thank my research partners from the industry for their contributions to my research, funding, and data that made this research project possible. I thank Christian, Martin, Ralf, Richard, Rolf, Tobias, and Ursula from the case study companies. Among others, I also thank Alan, Alexios, David,

Lukas, and Mark for the insightful expert interviews. I thank BMBF's (Federal Ministry of Education and Research) Software Campus 2.0 project for the partial funding of my research. I also thank the researchers and industry experts I met at numerous conferences and events, including Bitkom, CROSS, EuroPLoP, HICSS, OODACH, OpenSym, and OSS.

Last but not least, I thank Shushan for her support and inspiration. Among others, I also thank my friends Arshak, Basti, Ben, Bettina, Ernst, Henrik, Hovo, Jake, and Katha.

1

Introduction

Companies have been using open source tools for software development for a long time [40] [46] [126] [125] [88], but in recent years more and more companies have been introducing open source components into their products with an estimate of above 90% of all commercial software including open source software [47]. While beneficial, this carries certain risks if a company has no rules or guidelines for such use of open source components.

The ungoverned open source use can be problematic to companies, if there is no defined pro-

cess for checking and documenting the components and their metadata before use. One common problem is non-compliance with open source licenses. Often software developers use open source components freely available on the web (e.g. on GitHub.com) without checking the licenses of the source code. While open source software is free, it does come with usage rules, that is licenses. Certain open source licenses require component users to open source (publish and use appropriate license) the software created using the components licensed under these licenses (e.g. GNU General Public License (GPL)). This concept is called copy-left effect and such licenses are referred to as copy-left licenses. Not complying with copy-left licenses can get the user sued, which has happened in the past with court decisions in favor of the open source license enforcers.

A recent example in the aerospace industry was the case of CoKinetic Systems Corporation[1]. One of the allegations was that Panasonic willfully violated GPL v2 open source license requirements. This comes to show that even market leaders like Panasonic, with about 70% share of the embedded in-flight entertainment hardware market, need to consider threats of open source license non-compliance and establish open source governance processes. The costs of being threatened with lawsuits include legal costs, reputation loss, and wasted management attention. If a lawsuit is filed, the company could be forced to reveal undesired information about its products and know-how. It can also be forced to recall its products that violate certain open source license requirements [129] [131]. The threats can translate into actual costs and losses for a company, including:

- Legal costs (litigation-related costs, lawyer fees, etc.)
- Compliance costs (costs of establishing compliance under time pressure)
- Financial costs (license and other fees, potential punitive damages)
- Production costs (cease and desist, product recall, etc.)
- Reputation loss / damage
- Undesired revealing of information (source code of the core product)

[1]From Web Archive – CoKinetic Systems, Corp. files Monopolization Claim Against Panasonic Avionics Corporation; Seeks Damages In Excess of $100 Million

- Increased risk of more lawsuits, and other risks.

We suggest corporate open source governance as a measure to mitigate and prevent the above-mentioned threats and risks. To mitigate these risks companies must govern their use of open source software thought FLOSS governance processes and guidelines.

We define FLOSS governance as the set of processes, best practices, and tools employed by companies to govern the use of FLOSS components as parts of their commercial products while minimizing their risks and maximizing their benefit from such use [68]. In the context of this dissertation, the definition of FLOSS governance should not be confused with other definitions that cover the governance of open source communities or projects, such as the definition by Markus [107]: "[Open source governance is defined as] the means of achieving the direction, control, and coordination of wholly or partially autonomous individuals and organizations on behalf of an open source software (OSS) development project to which they jointly contribute".

FLOSS governance can apply to the commercial use, contribution or leadership of open source projects. However, we limited the scope of this dissertation to the corporate use of open source components only, intentionally excluding governance considerations of companies contributing to or leading open source communities or projects. This focus enabled us to build and present a detailed theory covering multiple aspects of getting started with open source governance in companies, a topic of the highest practical relevance to most companies today and novel to FLOSS research [72].

Depending on the maturity of a company's open source governance, it can cover topics such as governance management, open source program office, license compliance, component search, component selection, component approval, component integration, component repository and reuse, software product model, supply chain management, communication, capabilities and more. In this project, we touched on several best practice categories of FLOSS governance, while maintaining our focus on supply chain management. We choose these topics as they are both novel in the research community and of high industry significance. The topics we considered are based on our review of the related literature, presented in detail in Chapter 2, and from the initial analysis of the expert

interviews, we conducted as part of theory building, presented in detail in Chapter 3. Some of the FLOSS governance topics and subtopics we identified include:

- Getting Started with Open Source Governance (transition to governance and risk analysis)
- Inbound Governance (component search and component approval)
- Supplier Management (bill of materials management and supplier audit)
- Outbound Governance (license compliance and release management)
- General Governance (open source program office and governance strategy).

Given the breadth of the topic of FLOSS governance, we limited the scope of this three-year research project to cover a mix of basic, intermediary and advanced subtopics. We focused our study on:

- Getting Started with Open Source Governance
- Inbound Governance
  - Component Approval
  - Component Repository
- Supply Chain Management
- Outbound Governance
- General Governance.

To cover the whole space of the issues on corporate open source governance emerging from the state-of-the-art review, we formulated the following research questions:

- *RQ-TB1: How should companies using open source components in their products get started with open source governance based on existent industry best practices?*
- *RQ-TB2: How should companies using open source components in their products govern the inbound aspects of the FLOSS use?*

- *RQ-TB3: How should companies using open source components in their products govern their software supply chains?*
- *RQ-TB4: How should companies using open source components in their products govern the outbound aspects of the FLOSS use?*

To answer the research questions, we employed the research method of qualitative survey [77]. This dissertation presents a qualitative survey exploring the open source governance experiences of the 21 experts we interviewed from the 15 companies that use open source components in their products and have an advanced understanding of open source governance processes and practices, as well as other primary materials (e.g. FLOSS governance guidelines, white papers). We performed data gathering and analysis using formal semi-structured interviews, researcher notes, and materials review. We interviewed FLOSS governance experts from 15 diverse companies chosen through a theoretical sampling of more than 140 companies in our network.

The contribution of the dissertation is a theory of industry best practices for corporate open source governance, presented in full detail in Chapter 3. The theory proposes a number of best practices we identified in the following thematic areas:

- Getting Started with Open Source Governance
  - Product Analysis (OSGOV-GETSTA-PROANA) - 8 best practices
  - Transition Organization (OSGOV-GETSTA-TRAORG) - 8 best practices
  - Transition Policy (OSGOV-GETSTA-TRAPOL) - 3 best practices
  - IP-at-Risk Analysis (OSGOV-GETSTA-IPRISK) - 9 best practices
  - Communication and Capabilities (OSGOV-GETSTA-COMCAP) - 5 best practices
- Inbound Governance
  - Component Approval (OSGOV-INBGOV-COMAPP) - 13 best practices
  - Component Reuse (OSGOV-INBGOV-COMREU) - 19 best practices
- Supply Chain Management

  - Supply Chain Management Policy (OSGOV-SUCHMA-SCMPOL) - 3 best practices
  - Supply Chain Management Process (OSGOV-SUCHMA-SCMPRO) - 5 best practices
  - Preventive Governance (OSGOV-SUCHMA-PREGOV) - 4 best practices
  - Corrective Governance (OSGOV-SUCHMA-CORGOV) - 4 best practices
  - Bill of Materials Management (OSGOV-SUCHMA-BOMMAN) - 4 best practices
  - License Compliance for Supply Chain (OSGOV-SUCHMA-LICCOM) - 2 best practices

- Outbound Governance
- General Governance.

Each of these themes covers a subset of industry best practices from our theory. We cast the individual best practices in the format of context-problem-solution patterns that, when combined, form a practical handbook of getting started with FLOSS governance in companies. The handbook section on getting started with FLOSS governance is presented in Appendix A, while that on supply chain management is presented in Appendix B. This format is well structured allowing for interconnection of best practices within and across the thematic subsections. At the same time, this format is actionable and highly practice-relevant, as it enables the industry to apply the findings of our research by adapting and applying the best practices we identified in their companies. The latter was a priority of ours, as we wanted to increase the potential impact of our work.

The proposed theory covers multiple aspects of corporate open source governance based on the qualitative data analysis of the expert interviews conducted over the course of three years. Beyond proposing a theory for corporate open source governance, we went on to evaluate it through case studies, presented in detail in Chapter 4. As the theory building was an iterative process, so was the theory evaluation. The early focus of our theory building was on *RQ-TB1: How should companies using open source components in their products get started with open source governance based on existent industry best practices?*. We answered RQ-TB1 through the part of our theory addressing industry best practices for getting started with corporate open source governance, described in

detail in Section 3.4.1 of Chapter 3. We then cast our theory as a set of best practice patterns, which became the first section of our open source governance handbook. See this handbook section in Appendix A. To evaluate the getting started part of our theory, we conducted a case study following Yin [157]. We chose a large German company operating internationally in four software-intensive industries, and using open source software in its products, as the subject of this case study. The company asked to be anonymous, so in this dissertation, we call it Company A that was the subject of Case Study A. First, we assessed the current use of open source at Company A and its state of FLOSS governance. We then guided the implementation of our FLOSS governance handbook's section on getting started, which enabled us to evaluate our proposed theory in a real-life environment at a company using open source components in its products. Case Study A continued for 2.5 years and enabled us to implement and evaluate further parts of our theory, namely focused on the inbound aspects of corporate FLOSS governance, such as component approval and component reuse.

In parallel to evaluating the getting started part of our proposed theory within Case Study A, we continued the theory building process addressing RQ-TB2 (on inbound aspects of FLOSS governance), RQ-TB4 (on outbound aspects of FLOSS governance). Once we completed our analysis of the inbound aspects of FLOSS governance, we extended our handbook with the industry best practices on open source component approval and component reuse. We then guided the implementation of a subset of best practices on inbound governance at Company A, which constituted the second phase of Case Study A. We concluded our theory building by answering RQ-TB3 (on software supply chain aspects of FLOSS governance), which is a more advanced aspect of corporate open source governance, as it deals with open source-related issues caused by different actors in software supply chains often beyond the control of the company on the end of a given supply chain. Best practices on this topic cover preventive and corrective measures of supply chain governance, license compliance within supply chains, and management of bills of materials. As Company A was just getting started with FLOSS governance, we chose another company – Company B – for

our second case study – Case Study B. We guided the implementation of supply chain management best practices at Company B to evaluate the core part of our theory. Case Study B went on for one year and enabled us to test the part of the proposed FLOSS governance theory on software supply chains. As a result of our theory evaluation, we found out that some parts of the proposed theory are lacking comprehension, completeness, and applicability. We reported these findings for each case study in Chapter 4.

Future research can both further extend the scope of the proposed open source governance theory and further evaluate the current theory.

1.1 Publications

In the course of our three-year research project on corporate open source governance, we published partial results of the study. In 2017, we outlined in a book chapter the state of the art of open source license clearance in software product governance, as an aspect of FLOSS governance of high industry [128]. We proposed some questions companies should ask when dealing with FLOSS governance in software product lines. We published another paper discussing the industry requirements for FLOSS governance tools, which is another key aspect of the corporate open source governance, as the use of open source cannot be efficiently governed without appropriate tools in accordance to the needs of the companies using open source software in their products [68] [69]. We wrote papers presenting parts of our proposed theory on open source governance, namely on getting started with FLOSS governance in companies [70], as well as on component reuse and its governance implications [71] in two different outlets. These papers are currently in review.

We plan to continue publishing further results of our study. We are planning to further publish our extensive literature review on corporate open source governance, whose highlights we presented in Chapter 2 in this dissertation. We will publish another paper with the extensions of the theory of industry best practices for FLOSS governance, whose highlights we presented in Chapter 3 in this dissertation. Finally, we will publish our complete theory evaluation based on the conducted case

studies, whose highlights we presented in Chapter 4 in this dissertation.

1.1.1 License Clearance in Software Product Governance

In 2017, we published a book chapter with our initial take on FLOSS governance and supply chain management [128]. While most software products include OSS components, the obligations that open source licenses put on their users can be difficult or undesirable to comply with for companies. Therefore, software vendors and related companies need to govern the process by which open source components are included in their products, which includes open source license clearance – how a company decides whether a particular component's license is acceptable for use in its products or not. In this article, we discussed this process, reviewed its challenges to companies, and provided unanswered research questions.

1.1.2 Understanding Industry Requirements for FLOSS Governance Tools

In 2018, we published a conference paper at the 14th International Conference on Open Source Systems with our theory on industry requirements for FLOSS governance tools [68]. As companies 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. We found that 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 investigated and presented common industry requirements for FLOSS governance tools, followed by an evaluation of the suggested requirements by matching them with the features of existing tools. The research method we employed was the QDAcity-RE method for structural domain modeling using qualitative data analysis [81].

1.1.3 Industry Requirements for FLOSS Governance Tools to Facilitate the Use of FLOSS Components in Commercial Products

In 2019, we submitted a journal paper with an extended theory industry requirements for FLOSS governance tools to the Journal of Systems and Software [69]. This was our follow-up to the conference paper *Understanding Industry Requirements for FLOSS Governance Tools*. This paper investigated and presented a more complete list of industry requirements for FLOSS governance tools including tool requirements for searching OSS components, selecting the most appropriate ones, detecting and preventing security vulnerabilities in OSS components, documenting and communicating a company's FLOSS governance strategy, checking for export restrictions, and other requirements. We also extended our theory evaluation to increase the rigor of our study. The research method we employed was the QDAcity-RE method [81].

1.1.4 Getting Started with FLOSS Governance and Compliance: A Theory of Industry Best Practices

In 2019, we submitted a conference paper to the 15th International Symposium on Open Collaboration with the first part of our theory on industry best practices for open source governance, focused on how companies should get started with corporate FLOSS governance. The paper was accepted and will be published in August 2019 [70]. As the commercial use of OSS is on the rise many companies lack FLOSS governance processes and are open to the risks of the ungoverned use of open source in their products, including legal, financial, intellectual property, and other risks. To mitigate these risks, companies are looking at getting started with governance. We interviewed industry experts to identify the state-of-the-art best practices on how to start governing the OSS use in a company. We also studied practitioner reports on existing practices for introducing FLOSS governance processes. We presented our findings in the actionable format of best practices patterns covering the transition to governance, we well as product and risk analysis. The research method we employed

was the qualitative survey research by Jansen [77].

1.1.5 Industry Best Practices for FLOSS Governance and Component Reuse

In 2019, we submitted a conference paper to the 23rd European Conference on Pattern Languages of Programs with another part of our theory on industry best practices for open source governance, focused on how companies should reuse the open source components they have previously used, as well as the governance practices and processes related to OSS component reuse and repositories. The paper was accepted and will be published in July 2019 [71]. Based on our expert interview, we identified that the industry best practices on the topic included the definition of a component reuse policy, operationalization of the component reuse policy in a component reuse process, creation and maintenance of a searchable component repository with a single well-defined location, guidelines for auditing and using the component repository, and integration of component reuse processes with other aspects of open source governance. The research method we employed was the qualitative survey research by Jansen [77].

1.2 Dissertation Structure

Chapter 2 presents the state of the art of open source governance research in the academic literature. We present the literature review of 87 publications on the topic of open source in general and on its specific aspects. We conduct the literature review following the method by Webster and Watson [149] and employing qualitative data analysis techniques and tools. We present the overview of the studied papers in Table 2.1. We present the code system for the qualitative data analysis conducted during literature review in Section D.1 in Appendix D.

Chapter 3 presents the theory of industry best practices for open source governance we developed following the qualitative survey method by Jansen [77]. This chapter covers the main contribution of the dissertation consisting of industry best practices on open source governance in general and

on its specific aspects presented in the subsections of the chapter. Subsection 3.4.1 covers industry best practices for getting started with open source governance in companies. Subsection 3.4.5 covers industry best practices for general governance. Subsection 3.4.2 covers industry best practices for inbound open source governance. Subsection 3.4.4 covers industry best practices for outbound open source governance. Finally, as the focal aspect of the inbound governance, subsection 3.4.3 covers industry best practices for supply chain management in terms of open source governance. While this chapter covers the proposed theory of industry best practices for open source governance, we also developed a practical handbook for open source governance that includes the derived best practices from the theory cast as a collection of interconnected patterns. Appendix A presents the subset of best practice patterns for getting started with FLOSS governance. Appendix B presents the subset of best practice patterns for supply chain management. Additionally, we present the code system for the qualitative data analysis conducted during the theory building qualitative survey in Section D.2 in Appendix D.

Chapter 4 presents the theory evaluation we conducted following multiple-case case study research method by Yin [157]. Taking parts of the proposed theory from Chapter 3, we selected three case study companies for the guided implementation and evaluation of the proposed industry best practices in the production setting. The companies, anonymized per their request, correspond to Case Study A, Case Study B, and Case Study C. Case Study A was a 2.5-year study into a large German company operating internationally in four software-intensive industries, and using open source software in its products. Company A had limited understanding of FLOSS governance, just getting started with governance, which made it a good fit for us to evaluate our theory's part on getting started with open source governance, as well as a subset of industry best practices on inbound governance. Case Study B was a one-year study into another large German company operating internationally in the enterprise software industry, and extensively using open source software in its products. Company B had an advanced understanding of the basics of FLOSS governance, and internal institutional support for governance processes. However, the company lacks open source

governance practices and processes for managing its software supply chains, which made it a good fit for us to evaluate our theory's part on supply chain management in terms of open source governance. In both cases, we followed a case study protocol we developed following Yin [157], which is presented in Appendix E. In both case studies, we started by assessing the current situation in regards to corporate open source governance conducting more than 20 situation assessment interviews with the employees in different divisions of Companies A and B. Subsection 4.4.1 presents the initial situation assessment at Company A. Subsection 4.5.1 presents the initial situation assessment at Company B. We then guided the implementation of the selected parts of our open source governance handbook in Companies A and B, followed by the observation and evaluation of companies' use of the proposed industry best practices from our theory in production-level projects. Finally, we present our evaluation for Case Study A in Subsection 4.4.2, Subsection 4.4.3, and our evaluation for Case Study B in Subsection 4.5.2.

Chapter 5 concludes the dissertation by reassessing the main contributions of our work, including the state of the art, the proposed theory, and its evaluation, as well as the potential directions for further research on open source governance. We also discuss the limitations of our study both in terms of methods employed, and data collected.

2

State of The Art

This chapter covers the current academic literature on corporate open source governance. We report on the literature survey we have conducted analyzing 87 academic papers on the topic. First, we cover the potential risks of open source use in products. Addressing these risks, we then focus on corporate open source governance broadly. Finally, we discuss the state of the art of the specific aspects of governance, including that of our focal topic – supply chain management. We compare and contrast the analyzed state of the art literature with the insights and industry best

practices from our theory.

2.1 Overview

Open source software and open source development have been extensively researched [34] [79], but only little research focused on the topic of corporate open source governance in particular (unlike open source community governance [118] [117] [95], for example). After an initial literature search and familiarization with the topic, we undertook a systematic review of the related work to discover the key concepts of corporate open source governance previously studied by peer researchers.

In the course of the literature survey, we started by analyzing the main motivation for corporate open source governance – the potential risks of open source use in products. In Section 2.4, we presented the state of the art findings on the latter, while referencing the industry best practices from our theory.

Analyzing the collected papers employing a qualitative data analysis (QDA) tool called MAXQDA[1], we then identified the core topics of corporate open source governance in the state of the art literature:

- Getting Started – *GT*
- Inbound Governance – *IG*
- Outbound Governance – *OG*
- General Governance – *GG*.

The literature on *Getting Started with Open Source Governance* focused on the transition from ungoverned use of open source in companies to institutionalized governance, as well as building a product architecture mapping the previously used open source software.

The literature on *Inbound Governance* focused on managing how open source software would get into the company (and its products). Inbound governance addressed engineering management,

[1] MAXQDA – Qualitative Data Analysis Tool – https://www.maxqda.com/

open source component selection, approval, and integration, as well as supply chain management – *SCM*. The latter was such a large subtopic on its own that we split it from the overarching inbound governance topic. SCM addressed the governance issues of BOM management, quality management of the supplied code, supplier certification and standards.

The literature on *Outbound Governance* focused on the external aspects of corporate open source governance, such as license compliance for the shipped products that included open source software, and product release.

The literature on *General Governance* focused on the aspects of corporate open source governance that did not fit into one of the above-mentioned categories. This topic included a variety of concepts that affect all the other aspects of open source governance. General governance addressed open source program office, governance related education and communication, and governance management.

To review the state of the art in the research community on corporate open source governance, we undertook a systematic and comprehensive approach by conducting a literature survey. We followed the literature review methodology by Webster and Watson [149] identifying and studying 87 papers. These papers include 44 peer-reviewed journal papers, 30 peer-reviewed conference papers, seven peer-reviewed workshop papers, as well as six technical reports. We conducted a qualitative data analysis of these papers, which resulted in a code system of key governance concepts from the literature. We mapped the papers and the top-level governance topics presented above, whose overview you can see in Table 2.1 (papers ordered by the year of publication). See Section D.1 in Appendix D for the full code system of the concepts emerging from the QDA.

ID	Year	Type	Publication Outlet	GS	GG	IG	OG	SCM
[96]	2019	Journal	International Free and Open Source Software Law Review (IFOSSL Review)				X	
[32]	2018	Journal	IFOSSL Review		X		X	X
[51]	2018	Conference	Supporting Groupwork					X

ID	Year	Type	Publication Outlet	GS	GG	IG	OG	SCM
[151]	2018	Conference	European Conference on Pattern Languages of Programs (EuroPLoP)			X	X	
[50]	2017	Conference	International Symposium on Open Collaboration (OpenSym)	X		X	X	
[136]	2017	Conference	OpenSym	X	X		X	
[44]	2016	Conference	Computer Software and Applications		X	X	X	
[49]	2016	Conference	IFIP International Conference on Open Source Systems (OSS)	X				
[100]	2015	Journal	Data and Knowledge Engineering			X	X	
[135]	2015	Conference	Computer Law Review International			X	X	X
[142]	2015	Conference	Practice of Enterprise Modeling		X			
[129]	2014	Report	FAU University Erlangen–Nürnberg		X	X	X	X
[33]	2013	Conference	OpenSym	X				X
[47]	2013	Conference	Software Technologies					X
[101]	2013	Journal	IFOSSL Review				X	X
[102]	2013	Conference	OSS			X		
[5]	2012	Journal	Information Management	X			X	
[53]	2012	Journal	IEEE Software	X	X	X	X	
[55]	2012	Workshop	IT Project Management	X	X			X
[8]	2011	Journal	IEEE Software	X				
[60]	2011	Conference	Artificial Intelligence and Law	X			X	
[84]	2011	Journal	IFOSSL Review	X		X	X	X
[144]	2011	Journal	Association for Information Systems		X			
[6]	2010	Conference	Software Business			X	X	
[25]	2010	Conference	E-Business Engineering	X	X			

ID	Year	Type	Publication Outlet	GS	GG	IG	OG	SCM
[31]	2010	Journal	IFOSSL Review	X	X	X	X	X
[39]	2010	Workshop	Emerging Trends FLOSS Research and Development (FLOSS)				X	
[63]	2010	Conference	Academic MindTrek	X				
[72]	2010	Journal	Information and Software Technology		X	X		
[92]	2010	Workshop	Free/Libre Open Source		X	X		
[98]	2010	Journal	EuroPLoP				X	
[104]	2010	Journal	Information Systems		X			
[113]	2010	Journal	Advances in Information Systems	X	X			
[134]	2010	Journal	Open Source Software and Processes	X	X	X		
[140]	2010	Journal	IFOSSL Review				X	X
[141]	2010	Workshop	FLOSS			X		
[155]	2010	Journal	IFOSSL Review	X	X		X	
[4]	2009	Workshop	FLOSS	X		X	X	X
[43]	2009	Journal	R&D Management				X	
[54]	2009	Conference	International Conference on Software Engineering (ICSE)	X		X	X	
[78]	2009	Conference	ICSE		X	X		X
[82]	2009	Journal	IFOSSL Review	X	X	X	X	X
[91]	2009	Journal	Management Science			X		
[115]	2009	Conference	South African Institute of CS and IT		X			
[137]	2009	Conference	Americas Conference on Information Systems	X	X		X	
[139]	2009	Journal	Research Policy				X	
[143]	2009	Journal	IEEE Internet Computing			X		

ID	Year	Type	Publication Outlet	GS	GG	IG	OG	SCM
[10]	2008	Journal	Operations and Supply Chain Management	X				X
[15]	2008	Conference	Information Systems for Crisis Response and Management		X	X		X
[40]	2008	Conference	OSS			X		
[58]	2008	Conference	Mining Software Repositories	X		X	X	
[73]	2008	Conference	OSS	X		X		
[108]	2008	Journal	Information Economics and Policy				X	
[14]	2007	Journal	Comparative Economics				X	
[90]	2007	Journal	Transactions on Software Engineering			X		
[107]	2007	Journal	Management and Governance	X	X	X		
[117]	2007	Journal	Management and Governance		X	X		
[118]	2007	Journal	Academy of Management		X			
[125]	2007	Journal	IEEE Computer				X	
[87]	2007	Journal	Strategic Information Systems	X		X		
[153]	2007	Report	Cambridge University					X
[158]	2007	Journal	Law, Commerce, and Technology				X	
[159]	2007	Journal	Cyber Law	X		X	X	
[17]	2006	Journal	Management Science	X				
[46]	2006	Journal	MIS Quarterly	X	X	X		X
[103]	2006	Conference	OSS			X		
[106]	2006	Conference	Automated Software Engineering			X		
[21]	2005	Journal	IBM Systems	X	X		X	
[35]	2005	Journal	Research Policy				X	

ID	Year	Type	Publication Outlet	GS	GG	IG	OG	SCM
[57]	2005	Conference	International Symposium Empirical Software Engineering		X	X		
[93]	2005	Journal	Economic Perspectives	X			X	
[16]	2004	Report	University of Lincoln			X		
[37]	2004	Conference	Hawaii International Conference on System Sciences	X			X	
[131]	2004	Journal	IEEE Software	X	X	X	X	
[154]	2004	Report	San Jose State College of Business	X		X		
[18]	2003	Journal	Research Policy	X				
[36]	2003	Workshop	Standard Making	X				
[75]	2003	Journal	Organization Science	X				
[152]	2003	Journal	Research Policy	X				
[19]	2002	Conference	Software Reuse	X	X	X	X	
[28]	2002	Report	Carnegie Mellon University		X	X		
[89]	2002	Report	Stanford Institute for Economic Policy Research	X	X			
[116]	2002	Workshop	Principles of Software Evolution			X		
[12]	2001	Conference	Computer Documentation	X	X	X		
[83]	2001	Journal	Oxford Review of Economic Policy	X	X	X	X	
[120]	2000	Journal	Computer Law and Security Review	X			X	
[124]	2000	Conference	Application of Intelligent Agents	X		X		

Table 2.1: Literature Survey – Papers on Corporate Open Source Governance

2.2 Research Question

To define the scope of the state of the art review of corporate open source governance, we asked the following overarching research question (Research Question – State of the Art):

RQ-SA: How do companies conduct open source governance to address the potential risks of using open source software in their products?

We broke down the research question into the following components:

- potential risks of using open source software in their products
- corporate open source governance as a way to address the risks
- central aspects of corporate open source governance
- recommended best practices from the literature.

Our goal at this stage was to conduct a comprehensive literature review identifying the key motivation for companies establishing FLOSS governance – the legal, technical, and business risks caused by the ungoverned use of open source in products. Our next goal was to study the state of the art literature on the potential solutions to such risks. Our preliminary literature review indicated that corporate open source governance could be such a solution. Reviewing the related literature we aimed at identifying the central aspects of FLOSS governance, which would inform our theory building giving a frame of comparison and an academic basis for our theory. Finally, we aimed at identifying any recommended best practices by the research community for the corporate open source governance. We planned to compare and contrast the best practices from our theory to those from the literature review.

2.3 Research Method

To answer the research question RQ-SA, we followed the literature review method by Webster and Watson [149], gathering 87 papers on the topic of open source governance in companies. We then

analyzed these papers through a qualitative data analysis (QDA) process assisted by a QDA tool – MAXQDA. In the course of the QDA, we identified the common concepts across the papers, documented and defined them in a QDA code system, which we used in our iterative analysis. During the iterations, we added new papers, applied the defined codes, and modified the codes to best capture the common themes in the state-of-the-art review. Finally, we split the codes that were encompassing multiple subthemes, merged the ones that could be better defined as single unified codes, and removed some of the initially identified but not often used codes. The final code system and the overview of the coded segments are presented in Appendix D.1.

Following the literature review method by Webster and Watson [149], we took the following steps:

- *Step 1.* General Literature Search
- *Step 2.* Focused Literature Search
- *Step 3.* Backward and Forward Literature Search
- *Step 4.* Literature Summarization
- *Step 5.* Defining Unit of Analysis
- *Step 6.* Qualitative Data Analysis of Literature
- *Step 7.* Reporting State of the Art for Core Concepts.

Steps 1, 2, and 3 focused on searching and collecting the relevant literature on the risks of ungoverned OSS use and on corporate open source governance. In this early stage, we used different keywords to search scientific databases and search engines to identify the potential papers for our analysis. Some of the keywords we used were:

- open source software, OSS, open source components
- open source use, open source usage, OSS use, FLOSS use, best practices for open source
- governance risks, open source governance risks, open source license compliance risks, open source legal risks, open source supply chain risks

- open source governance in companies, corporate open source governance, best practices for open source use risks, open source usage risks, corporate risks of open source use, best practices for open source governance.

We also used the keywords in different combination (e.g. open source use risks, open source use governance, open source program office best practices, etc.).

The main platforms / libraries we searched included:

- Google Scholar
- Web of Science
- Scopus
- ACM Digital Library
- EBSCO
- ScienceDirect.

After the first round of general literature search, we conducted a focused search on the corporate open source governance topics that emerged from the preliminary analysis of the identified general papers. Some of the keywords we used in this focused search included:

- transition to open source governance, open source component search
- open source review, open source license scanning
- open source program office, open source in bill of materials
- open source governance methods, open source governance standards, etc.

Following Webster and Watson [149], we also conducted forward and backward search using the references included in the identified literature, and those of the identified papers in other literature. We collected the PDFs of all the papers and organized them per preliminary topic.

In *Step 4.* we summarized the collected literature writing 2-3-paragraph-long summaries for each paper. These summaries were used for the preliminary analysis of the papers that helped us prepare

for the comprehensive analysis using a QDA tool. We used these early summaries to outline the key topics in our QDA code system.

In *Step 5.* we defined the unit of analysis of our literature review. In our case, the key unit of analysis were the top-level topics of corporate open source governance, such as Getting Started, Inbound Governance, Supply Chain Management, etc. The more detailed units of analysis were the recommended best practices in each of these topical categories. In this step we deviated from the units of analysis recommended by Webster and Watson [149], rather focusing on the concepts of the topic and not the different views on them (e.g. organizational, group, individual view).

In *Step 6.* we added all the collected papers into the QDA tool – MAXQDA, which we used to create a code system of concepts traced to the text segments from the studied papers. During the QDA process, we started by open coding – deriving the codes and adding them to the code system, which was followed by axial coding – structuring the open codes into a hierarchy of concepts. We finished the QDA process by conducting selective coding – applying the structured codes in the final version of the code system to support the central concepts with further data traces. See the final version of the code system, the specific codes, and the number of codings for each code in Section D.1 in Appendix D.

In *Step 7.* used the findings from the qualitative data analysis to report the literature review result per top-level concept category. We presented highlights from select papers, their insights (e.g. recommended governance best practices from literature), and their comparison to our proposed theory and specific industry best practices.

In Section 2.4 we present our literature findings on the risks of open source use in companies. In Section 2.5.2 we present our literature findings on inbound open source governance. In Section 2.5.3 we present our literature findings on supply chain management governance. In Section 2.5.4 we present our literature findings on outbound governance. In Section 2.5.5 we present our literature findings on general aspects of corporate open source governance.

2.4 Risks of Open Source Use in Companies

Corporate use of open source software in products has a number of benefits to companies, such as OSS being quickly available, of high quality, and low cost, as well as the fact that open source components and standards are universally accepted, widely tested, highly secure and well maintained by professional communities. However, in out literature review, we found that there were a number of risks of the ungoverned use of open source software in products. Some risks and challenges of open source use in companies were covered in the related work by Ruffin and Ebert [131], Franch et al. [47], Stol and Ali Babar [141], Popp [123], Helmreich [74], and others. Some of the key challenges included dependency on the community of an open source project [27] [29] [86] [1], complex licensing [111] [76] [105], and low quality documentation [2] [7] [105] [1]. We also identified that the various issues and risks of using open source software commercially could be mitigated through open source governance processes and practices. Open source governance addresses, among other issues, license compliance management [52], and related tooling [68] [80].

Li et al. [94] conducted an empirical study to show that, among other issues, developers in companies often underestimated the integration efforts related to the use and governance of open source software. While using FLOSS component commercially can seem free of charge at the first glance, a deeper assessment of the corporate OSS use uncovers a set of indirect costs related to OSS component review, compliance, integration, and maintenance. In each of these steps companies need to understand the associated risks of open source use, must asses these risks and must govern their use accordingly. Our theory discovered that the industry experts we interviewed recognize these risks and actively work on preventing or mitigating them. Our best practices covered the risk assessment of using open source component, both on its own and in comparison to the alternatives, such as in-house developing and outsourcing. The best practice A.4.5 (*OSGOV-GETSTA-IPRISK-2. Analyze risk exposure of using an open source component*) suggested how companies should analyze the potential risks of the corporate use of open source software. We also derived how companies should mitigate such risks in the best practice B.4.2 (*OSGOV-GETSTA-IPRISK-3. Mitigate risk*

to intellectual property) and the detailed best practices it encompasses, including A.4.7 (*OSGOV-GETSTA-IPRISK-3.1. Replace problematic components*), A.4.8 (*OSGOV-GETSTA-IPRISK-3.2. Decouple problematic components*), A.4.9 (*OSGOV-GETSTA-IPRISK-3.3. Require bill of materials for supplied code by 3rd party post-factum*), and A.4.10 (*OSGOV-GETSTA-IPRISK-3.4. Run random audits to identify previously undetected or missed open source components and their metadata*).

Ruffin and Ebert [131] talk about possible risks and benefits of using open source software. Besides advantages like saving time and improving security, they point out that companies should be vigilant about the open source components used in their products, preventing possible copyright infringement of third parties and their intellectual property rights. They also talk about several actions that can be undertaken to mitigate legal exposure, such as governing the use of open source components through well-documented processes. Once the company's developers go through this process, they can reuse the once used components and check them into a component repository. We confirmed their findings and identified best practices that help avoid potential risks of the ungoverned use of open source components, as well as develop efficiently through component reuse. Namely, the best practice A.4.5 (*OSGOV-GETSTA-IPRISK-2 Analyze risk exposure of using an open source component*) covers the potential risks of using open source components and suggests how companies analyze and prevent such risks. Moreover, the best practices *OSGOV-INBGOV-COMREU-11. Audit component repository*, *OSGOV-INBGOV-COMREU-18. Add security check information to component repository* confirm these findings from related literature.

2.5 Open Source Governance in Companies

We found that, as a solution to the potential risks of the ungoverned use of open source software, companies need to establish and follow formal open source governance, which included the core concepts of getting started with FLOSS governance, inbound governance, outbound governance, and general governance. One specific topic within inbound governance was large and extensive on its own – supply chain management governance. In Figure 2.1 we present an overview of some of the

key concepts resulting from our literature review. In the subsections of this section, we go into the details of each of these categories.

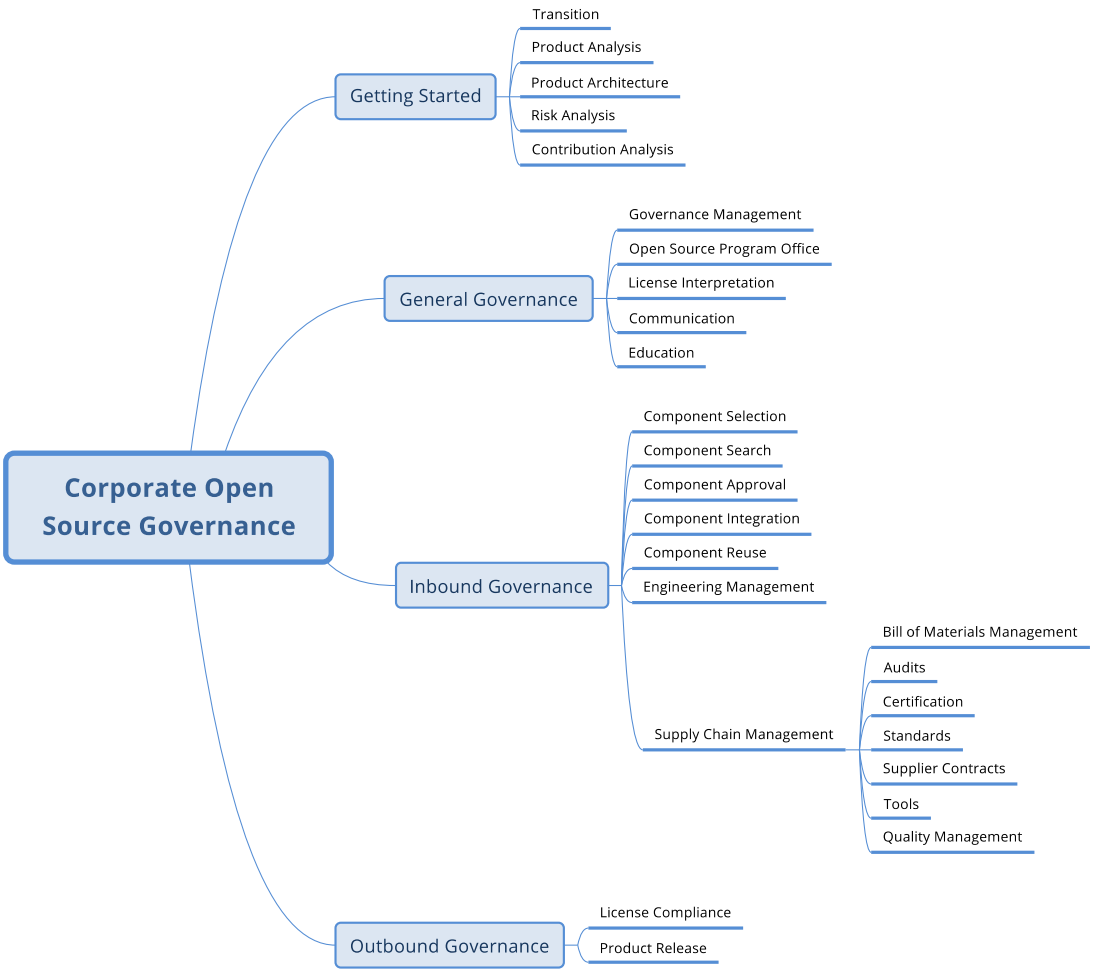


Figure 2.1: Main Concepts from Literature – Corporate Open Source Governance

2.5.1 Getting Started

Researchers have recognized many benefits of open source software adoption by companies including better interoperability, interconnectivity, trialability, transparency due to the availability of the source code [17] [26] [37] [46] [122] [125]. While some literature exists on FLOSS adoption in industry and on FLOSS governance in general [3] [18] [22] [57] [72] [123] [148] we found little re-

search particularly about industry best practices for getting started with open source governance in private companies. Therefore, we also reviewed general FLOSS governance research literature, and compared and contrasted it with our findings on the getting started aspects of the phenomenon.

Bonaccorsi & Rossi [18] discuss three key economic problems that arise with the emergence of the commercial use of open source: motivation, coordination, and diffusion. As they explain the different types of open source users, they introduce coordination as a basic alternative to open source governance including having a centralized leadership structure and a clear hierarchical organization or having technical support systems within a company to deal with the use of open source components. They discuss the diffusion of open source in companies. We do not discover industry best practices for a clear hierarchical organization when dealing with open source adoption. Instead, we find that the transition towards open source governance should involve stakeholders from all hierarchical levels in a company, guided by a transition policy outlined in A.2.1 (*OSGOV-GETSTA-TRAPOL-1 Establish FLOSS governance policy for the transition period*). This best practice also confirms findings by Lerner & Tirole [93] who highlight open source governance policies and internal legal systems as a way to prevent potential risks of ungoverned FLOSS use.

Kemp [82] talks about the operational aspects of the transition towards open source governance in companies. Analyzing the management perspective on the transition, he indicates that the goal for the management undertaking the transition is to install integrated processes across all relevant business functions to manage the effective use of FLOSS throughout the organization. He argues that in order to get there, an organization should consider disassembling the various pieces into their building block components and threading them together by start point (achievements to date), people (stakeholders) and the strategic, policy and process aspects. We find industry best practices matching Kemp's findings. Namely, the best practice A.3.9 (*OSGOV-GETSTA-PROANA-3.1 Run open source use analysis in products*) ensures that the management is aware of the various open source components that are currently used in the company, which leads to the best practice A.3.10 (*OSGOV-GETSTA-PROANA-3.2 Document current open source use*). As to the governance tran-

sition process, we identified the practice A.1.6 (*OSGOV-GETSTA-TRAORG-6 Establish the transition process*) that covers the operational aspects of setting up the process.

Fendt et al. [44] discuss some critical risks that can arise from the ungoverned use of open source software in products, such as compliance issues when dealing with open source software licenses. They go on to describe a suggested governance process and framework that aim to allow only appropriate FLOSS components into products, to protect internally developed code from potential risks of license non-compliance, and to assure the fulfillment of all license requirements. Furthermore, they state that the process automation and other factors have to be considered when implementing a FLOSS governance process. Our theory touches on some of these issues discovering, for example, an industry best practice for using tools to automate parts of the getting started process in companies – A.3.4 (*OSGOV-GETSTA-PROANA-1.3 Select and use governance tools for automation*).

Fitzgerald [46] talks about the specifics of product analysis as he describes the transformation from open source software development to an emerging commercially viable form of open source he calls OSS 2.0. He talks about the commercial use of open source software and the challenges of this transition. Our theory addresses some of these challenges providing industry best practices for the initial product analysis in particular. The best practices A.3.1 (*OSGOV-GETSTA-PROANA-1 Use a combination of methods for product analysis*) covers the methods a company can use for the initial product analysis to identify the previously used yet ungoverned open source components.

The topic of open source governance introduction in companies is of high practical relevance to the industry. Practitioners like Peters [122] analyzed some getting started aspect of open source governance. He highlighted the importance of open source governance policies during the transition to FLOSS usage in companies. He provided a guide intended to support the creation of a company's open source governance policy. He presented tips and best practices of writing such a policy, intended to regulate the use of open source in corporate environments. These best practices focused on identifying stakeholders, choosing a strategy, and setting the scope. Our theory confirms some of the best practices he identified. Namely, the best practice A.1.1 (*OSGOV-GETSTA-TRAORG-1*

Establish a board of stakeholders to organize the transition) sums up the need to identify the stakeholders interesting in the introduction of open source governance processes in the company, and presents the specifics of organizing these stakeholders. Another best practice in our theory A.1.4 (*OSGOV-GETSTA-TRAORG-4 Start small, then replicate - define the scope of the transition process*) deals with the scope of the transition towards open source governance.

Bonaccorsi et al. [17] talk about companies that use open source software in their products as part of their business strategy. They discuss how using open source influences a company's business model choice. They state that many companies choose to adopt a hybrid business model that comprises proprietary as well as open source products and services. Besides, they also talk about a company's motivation to use open source software in their products and about the factors that influence a company's openness towards using open source. Considering the scope of our paper, we did not investigate the influence of using open source on a company's business model, but rather focused on the reasons and techniques companies follow when getting started with FLOSS use in products. In line with Bonaccorsi's findings, our theory recognizes communication and capability building as central topics of industry best practices for open source governance. Namely, we identified a subset of industry practices on the issues including A.5.1 (*OSGOV-GETSTA-COMCAP-1 Establish communication channels for open source governance handbook*) to A.5.5 (*OSGOV-GETSTA-COMCAP-5 Provide employee training*), which talk about setting up internal communication channels, developing and providing employee training.

2.5.2 Inbound Governance

Researchers have studied different aspects of inbound open source governance, such as engineering management and software development [12] [131] [46] [72], open source component search [100] [53], open source component selection [6] [44], open source component approval [84] [57], open source component reuse [19] [53] and other subtopics of inbound governance. We present some highlights from our literature review on inbound governance in this subsection.

With the growing availability of high-quality FLOSS components, software developers increasingly use FLOSS components in commercial products, which among other things enables faster development and lower costs as illustrated in the *BOOTSTRAP* pattern by Weiss [150]. FLOSS governance policies in many companies require developers to track and document such FLOSS use [74] [123]. This enables the well-structured management and reuse of FLOSS components that have been added into product software. Umarji et al. [147] suggest using FLOSS governance tools to create and maintain repositories of reusable FLOSS components. Our findings confirm this in the best practice patterns *OSGOV-INBGOV-COMREU-8. Create component repository*, *OSGOV-INBGOV-COMREU-9. Update component repository*, *OSGOV-INBGOV-COMREU-10. Maintain component repository*, *OSGOV-INBGOV-COMREU-12. Use tools to create, update and maintain component repository*. Other studies focused on the risks of bill of materials management [51] [32] [101] [140], and maintenance of FLOSS component metadata [68]. Our theory confirmed and captured industry practices that addressed these risks.

Copenhaver [31] analyzed open source policies and processes focused on inbound governance. The author reported that companies set up a *Business Readiness Review (BRR)* process, in part, for open source component approval – originally proposed and developed by Carnegie Mellon West, O'Reilly CodeZoo, SpikeSource, and the Intel Corporation. In line with the BRR process, Copenhaver suggested the following criteria for open source component approval:

- functionality (how the software meets user requirements)
- usability (how intuitive, easy to install, easy to configure, and easy to maintain the software is)
- scalability (can the software cope with high-volume use)
- support (how many sources of support are available)
- documentation (is there good quality documentation)
- adoption (has the software been adopted by the community, the market, and the industry)
- community (is the community for the software active and lively)

- quality, security, etc.

Many of the above-mentioned criteria were confirmed by our theory of industry best practices for corporate open source governance, and its subtopic on open source component approval in particular. Namely, the best practice *OSGOV-INBGOV-COMAPP-4. Define transparent rules for open source component approval* addressed the criteria of functionality, adoption, community, support, etc.

2.5.3 Supply Chain Management

The reviewed literature addressed different aspects of corporate open source governance in software supply chains, such as the supply chain management policy and process [82] [4] [47] [15], BOM management [140] [101] [84] [55], supplier standards [153] [33], etc.

Germonprez et al. [55] presented the protection and compliance in open source supply chains in the context of FLOSS governance risk mitigation. The authors discussed potential governance and compliance risks companies could face because of the lacking governance of their software supply chains. As a recommended solution, the authors recommended proposed using open source component and license scanning tools (e.g. FOSSology) integrated with the bills of materials requested from the suppliers. The BOMs need to have a structured and standard format for the efficient supply chain management. The proposed format for BOM management is the software package data exchange (SPDX)[2] – a standard format for communicating the components, licenses, and copyright metadata associated with open source software packages. The SPDX standard could help facilitate compliance with free and open source software licenses by standardizing the way license information was shared across the software supply chain. In our theory, we confirm this insight, as most governance experts also highlighted the importance of managing BOMs in the context of SCM, as well as using BOM documentation and communication standards such as SPDX. In our proposed industry best practice B.4.10 (*OSGOV-SUCHMA-BOMMAN-4. Use machine-readable and standard*

[2] SPDX (software package data exchange) – https://spdx.org/

format for BOM upon software supply) we confirmed that companies should use a machine-readable format for its suppliers' bills of materials.

Kemp [82] discussed operational compliance of open source use in companies, focusing in particular on supply chain management policy. The author proposed that a SCM policy for open source governance should address:

- suppliers' open source compliance training
- automated code scanning to facilitate discovery and recognition of OSS in the supplied products
- procedure to prepare an open source bill of materials.

Kemp also recommended using The Linux Foundation's Self-Assessment Checklist to effectively assess supplier compliance practices and to engage suppliers in a discussion about compliance. Our theory's best practice B.3.2 (*OSGOV-SUCHMA-PREGOV-1.1. Assess open source governance and compliance awareness and maturity*) confirmed this recommendation proposing the assessment of FLOSS governance maturity of the suppliers.

Blecken and Hellingrath [15] studied the software supply chain of the tools in the domain of humanitarian operation. They defined supply chain management as the integrated process-oriented planning and control of material, information and financial flows along the entire value chain from the customer to the raw material producer. Applying this definition to software development, open source components are similar to raw materials used in producing products. The authors suggested that the general objective of SCM should be to reduce supply chain inventory and order lead time and improve responsiveness and service levels. In the context of open source governance, our theory confirmed that the SCM policy should address similar issues. Namely, our theory's best practice B.1.1 (*OSGOV-SUCHMA-SCMPOL-1. Establish supply chain management policy*) proposed that a company would need to establish a supply chain management policy with the objective of reducing supply chain complexity by using open source governance standards for suppliers.

2.5.4 Outbound Governance

Some researchers studied outbound open source governance and its focal topics, such as ensuring the open source license compliance of the shipped products [96] [31] [135] [136], release review [100] [82] [155], patents and contributions [120] [50] [158], etc.

Researchers considered open source license non-compliance as one of the most critical risks of the ungoverned use of open source in products. A company could get exposed to such a risk if a product incorporating open source components were to be released without ensuring the open source license compliance first. Though, there are legal uncertainties around the enforcement of certain open source licenses, for example, the enforceability of the GPL (GNU General Public License) license in China discussed by Lin and Shen [96], we found that license compliance was the key topic of FLOSS governance within outbound governance. Copenhaver [31] reported several recommended practices for ensuring license compliance before product distribution. The first such practice focused on the main employee role dealing with this aspect of FLOSS governance – the Open Source Software Compliance Officer (OSSCO). The OSSCO in a large organization, for example, would perform tasks of an open source ombudsman maintaining some degree of a separation between the day-to-day business processes for outbound governance, being available to discuss concerns from individual employees concerned about the company's fulfillment of open source license obligations. On the other hand, in smaller organizations, the tasks of an OSSCO would focus on dealing with the day-to-day license compliance and release review. In our theory, we did not find an industry best practice focusing on the role of a compliance officer in particular, but rather captured a best practice for the Open Source Program Office to perform tasks of outbound governance. For example, our best practice 3.18 (*OSGOV-OUTGOV-LICCOM-1. Ensure license compliance*) presented the multitude of outbound governance tasks to be performed by the Open Source Program Office.

2.5.5 General Governance

Covering the crosscut and general topic of open source governance, researchers studied governance management [72] [21], governance strategy and policy [142] [104] [134] [115] [44] [118], open source program office [155] [137] [55] [25], etc.

One of the essential topics of general governance included corporate governance policies that cover the company's principles when using open source components, as well as working with open source communities and other companies on governance-relate issues. Lundell et al. [104], when discussing the open source use in Swedish companies, brought up one aspect of company policy for FLOSS governance – the alignment between open source communities and commercial software development organizations. The recommended policy suggested that such an alignment could promoter the change of perceptions, development processes, and business models in companies using open source components. Among other governance-related tasks, this would be the task of an open source program office (OSPO) at a given company. Another specific OSPO task in this context would include promoting open source development practices within companies, called inner source [23]. Though we did not find governance industry best practices on inner source, in particular, we did outline other tasks of an open source program office (OSPO), such as defining and implementing an open source governance policy across the company. Our theory's best practice 3.19 (*OSGOV-GENGOV-GOVMAN-3. Establish an open source program office*) addressed the establishment of an OSPO and its key tasks.

Koltun [84] suggested some other governance-related goals and responsibilities of the OSPO (he called it Open Source Review Board), such as reviewing OSS use in product context – as part of an product architectural diagram that would show how the software components (including OSS) interfaced and interacted with the rest of the product's software. The OSRB would also examine licensing implications of the architecture, compatibility of components from a licensing perspective, and resultant license obligations. Koltun concluded that an OSRB would need to incorporate the expertise of skilled software architects and licensing experts with direct insight into company

product development plans and history. Confirming this insight, the industry best practice 3.19 (*OSGOV-GENGOV-GOVMAN-3. Establish an open source program office*) suggested the following tasks an OSPO would need to perform:

- Define roles, responsibilities, and policies
- Provide roles, responsibilities, and policies in written form
- Match policies to actual risks
- Collaborate with legal counsel on license interpretation
- Track industry best practices and standards
- Network to learn from others
- Engage with the community, etc.

3

Theory of Industry Best Practices for Corporate Open Source Governance

This chapter covers the theory of industry best practices for corporate open source governance – the main contribution of this research project and of the dissertation. With the literature review as our basis, we set out to investigate how companies using open source in their products govern this use. We asked research questions on how companies get started with governance, how

they deal with the inbound and outbound aspects of governance, as well as on how they manage their software supply chains. To answer these questions, we conducted a qualitative survey based on the practice-based data we collected during the first two years of this study. In five sampling iterations, we chose 15 companies with an advanced understanding and experience in FLOSS governance. We then interviewed 21 governance experts at these companies. Our theory of industry best practices was based on the qualitative data analysis of these interviews and of the primary materials. As a result of our research project, we identified the major parts of the theory, and discussed them in detail. We also developed a handbook of corporate open source governance based on our findings casting them in the actionable format of best practice patterns. In this chapter, we presented examples of these state-of-the-art practices and their data traces to our analysis.

3.1 Overview

In this dissertation, we propose a theory of industry best practices for corporate open source governance consists of the following focal parts:

- *Getting Started*
- *Inbound Governance*
- *Supply Chain Management*
- *Outbound Governance*
- *General Governance.*

Getting Started is the first major part of our theory, which focuses on the industry best practices for getting started with FLOSS governance. Mainly targeting at software companies with little or no governance in place, this part of our theory reports our findings on how expert companies got started with FLOSS governance, including but not limited to the specifics of the transition process from ungoverned use of open source to initial governance, the review of the existing products and

the open source components used in these products, the subsequent risk analysis and mitigation. We present our theory's detailed take on Getting Started in Section 3.4.1.

Inbound Governance is another major part of our theory, which focuses on the industry best practices for the inbound aspects of FLOSS governance. Inbound governance covers the best practices for dealing with the open source components coming from outside of the company, covering OSS component search, component selection, component approval, and component integration. Other practices and processes in the scope of inbound governance include open source component reuse and repository, and component monitoring. In the scope of this dissertation, we cover two subtopics of inbound governance in full detail – Component Approval and Component Reuse. We present our theory's detailed take on Inbound Governance in Section 3.4.2.

Supply Chain Management (SCM) is the focal part of our theory, which focuses on the industry best practices for the supply chain management aspects of FLOSS governance. SCM governance covers the best practices for dealing with the suppliers who deliver software that includes open source components ensuring license compliance, BOM management and other supplier-related governance aspects (supplier contracts, supplier audits, etc.). Supply Chain Management includes best practices for both preventive governance and for corrective governance. The former addresses supplier selection, supplier certification, and contracts. The latter addresses supplier audits, risk assessment associated with the supplied code, and risk mitigation. As a special topic of SCM governance highlighted by the industry experts, we detail the best practices for bill of materials management, including tracking and identifying the supplied open source components and their metadata, as well as using machine-readable formats and standards. Finally, SCM governance also covers license compliance within software supply chains. We present our theory's detailed take on Supply Chain Management in Section 3.4.3.

Outbound Governance is another major part of our theory, which focuses on the industry best practices for the outbound aspects of FLOSS governance. Outbound governance covers the best practices for dealing with open source license compliance and release of the products that use open

source software. Outbound governance includes practices and processes for checking and ensuring FLOSS compliance before products are shipped to the customers. Outbound governance also addresses the management of employee contributions to open source communities, as well as other governance issues in relation to external parties (customers, certification bodies, regulators, etc.). We present our theory's detailed take on Outbound Governance in Section 3.4.4.

General Governance is a major part of our theory, which focuses on the industry best practices for the general aspects of FLOSS governance, not focusing on any of the other focal topics presented above. General governance covers the proposed best practices for the establishment and operation of an Open Source Program Office – a body within a company managing the day-to-day issues of open source governance and compliance. General governance also covers the top management tasks of ensuring the strategic (company-wide) governance. General governance addresses other general issues such as FLOSS governance capability assessment and building among employees. We present our theory's detailed take on General Governance in Section 3.4.5.

In this dissertation, we also present excerpts from the governance handbook section on Getting Started in Appendix A and on Supply Chain Management in Appendix B. See our previously published work with an excerpt from the governance handbook subsection on Component Reuse (as part of Inbound Governance) [71].

Taking the FLOSS governance concepts from the literature survey presented in Chapter 2 as our basis, we conducted a qualitative survey [77] employing qualitative data analysis (QDA), which resulted in the above-mentioned five topics of FLOSS governance and their subcategories that were captured in our qualitative code system. Our work builds upon and extends the state-of-the-art FLOSS governance concepts, whose overview was presented in Figure 2.1.

Figure 3.1 illustrates an overview of the main concepts from our theory on the industry best practices for corporate open source governance. We go into detail on each of these concepts in this chapter, presenting specific best practices and their links that constitute the proposed theory.

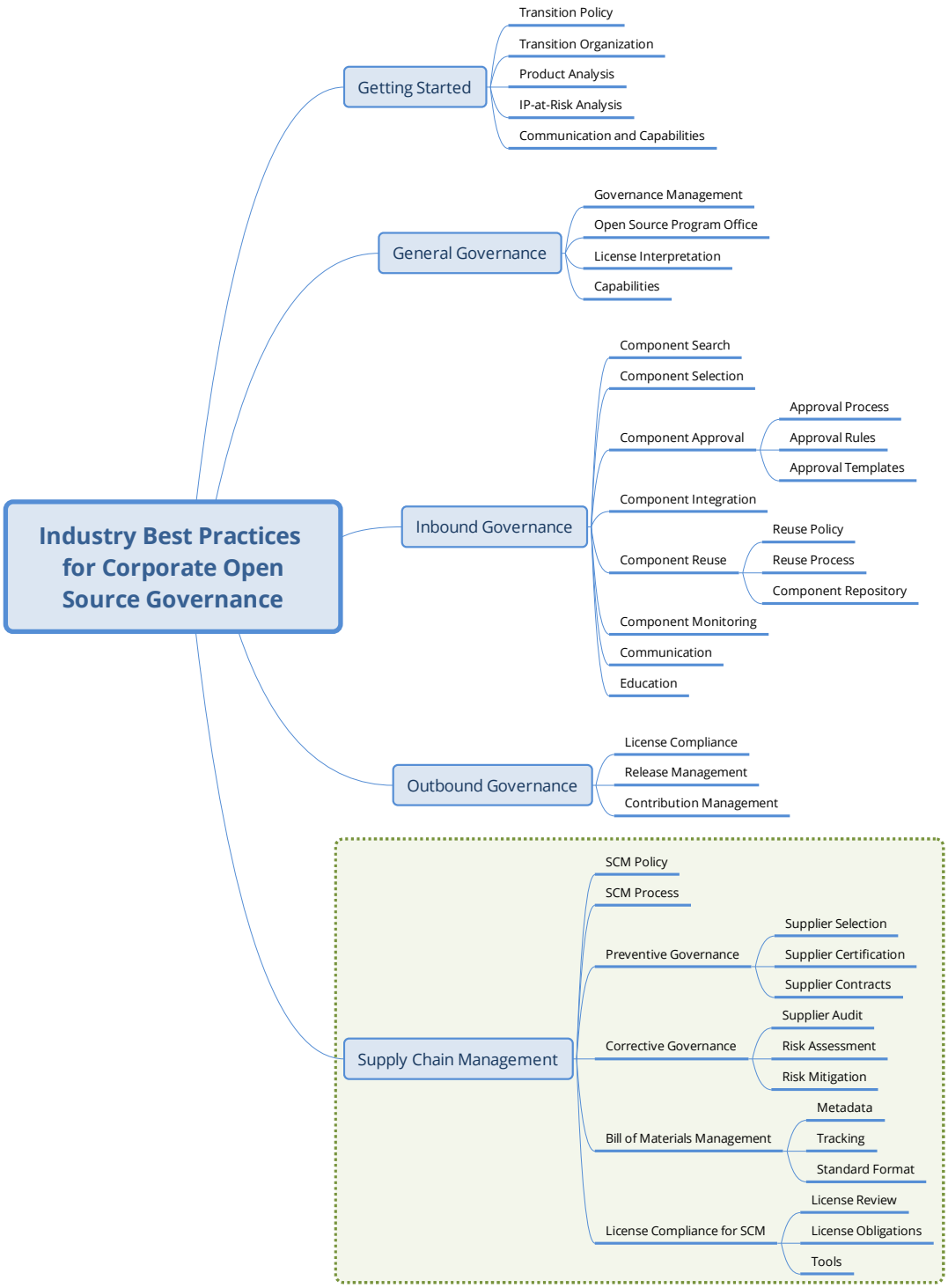


Figure 3.1: Industry Best Practices from Proposed Theory – Corporate Open Source Governance

3.2 Research Question

We started our theory building by asking broad, yet clear, focused, concise, complex, and arguable research questions. The questions we asked were based on our preliminary view of the potential scope of the theory that emerged from our state-of-the-art review. In asking the research questions we had two target audiences – academia and industry. We aimed at building a practice-based and practically applicable theory. Thus, we formulated exploratory and prescriptive research questions, which also affected our research method. To cover the whole space of the issues on corporate open source governance emerging from the state-of-the-art review, we formulated the following research questions:

- *RQ-TB1: How should companies using open source components in their products get started with open source governance based on existent industry best practices?*
- *RQ-TB2: How should companies using open source components in their products govern the inbound aspects of the FLOSS use?*
- *RQ-TB3: How should companies using open source components in their products govern their software supply chains?*
- *RQ-TB4: How should companies using open source components in their products govern the outbound aspects of the FLOSS use?*

Each of the above-mentioned four research questions was addressed during theory building and in our resulting theory. Beyond answering the four initial research questions, we found that some industry best practices emerging from our data did not directly answer any of the asked research questions. Such best practices were grouped together and presented as industry best practices for general governance.

While in the state-of-the-art review, we found supply chain management to be part of the inbound governance as the supplied code was one of the ways open source software would enter the company and become part of the company's products. However, as the topic of the most interest to

us and that with the highest perceived novelty and applicability, we dedicated an individual research question and section in the theory results for SCM governance.

3.3 Research Method

To answer our exploratory research questions we conducted a qualitative survey [77] [45] based on the data from open source governance expert interviews and primary materials on industry best practices for corporate open source governance. Methodologically, qualitative surveys resemble multiple-case case studies [77] [132] [157], in that they both are systems for collecting information from or about people to describe, compare, or explain their knowledge, attitudes, and behavior about a studied phenomenon in a real-life setting [45]. However, while case study research design enables an in-depth analysis of particular cases (usually a limited number of cases or a single case), the qualitative survey focuses on a less specific, yet more comprehensive and all-around perspective of the subject (usually from many data sources). As our goal was achieving the latter we used a qualitative survey to answer our research questions. Furthermore, the qualitative survey research method can be used to answer exploratory questions on emerging topics [77], which was the case in our study.

First, we set objectives to design and plan the qualitative survey, conduct a theoretical sampling in several iterations, choose expert interviews as our main source of data, collect data from expert interviews and primary materials, and analyze the collected data to build a theory of industry best practices for corporate open source governance. After defining our knowledge aims based on the state-of-the-art review and our research questions, we prepared the interview questions that covered different the predefined aspects and topics of open source governance emerging from our knowledge aims. We presented the details of the semi-structured interviews and interview questions in Section 3.3.2.

In parallel to conducting iterations of theoretical sampling, choosing expert companies, and interviewing experts from these companies, we also searched for primary materials on corporate open

source governance, including but not limited to company guidelines and policies for FLOSS governance, industry standards for FLOSS governance, recommended governance patterns by practitioners, white papers, slides, and on corporate open source governance.

We then transcribed and processed the interviews to prepare for data analysis. To analyze survey data from both the interviews and primary materials, we employed qualitative data analysis (QDA) aided by MaxQDA[1] (a QDA tool) in order to ensure the systematic analysis of the data and the traceability of our theory to the data. Following Jansen's method [77], we conducted an open (inductive) survey, in which the relevant topics of FLOSS governance, its dimensions and categories were identified through the interpretation of raw data – expert interview transcripts and primary materials. This approach was in contrast to the other type of qualitative surveys – the prestructured survey, in which some main topics, dimensions, and categories were defined before the study (e.g. industry best practices were already identified). In such surveys, the researchers would aim at finding these predefined phenomena in the research units, guided by a structured protocol for questioning or observation. The result of such a survey would not be an exploratory theory (which was our goal), but a descriptive analysis that would only show which of the predefined characteristics matched the empirical observations among the study participants [77].

Finally, we reported our findings as a theory of industry best practices in this dissertation. A best practice is a method reflecting the state-of-the-art as applicable in a particular context [127]. We further discussed this format of the theory presentation in Section 3.3.4 of this chapter.

3.3.1 Sampling

A qualitative sample should represent the diversity of the studied phenomenon (different topics of FLOSS governance) in the context of the target population (FLOSS governance experts in our case) [77] [45]. According to Jansen, researchers should purposely select a diverse sample with the aim to cover all existing relevant varieties of the phenomenon that would lead to theoretical saturation

[1] MAXQDA – Qualitative Data Analysis Tool – https://www.maxqda.com/

[77].

To build a theory of industry best practices for corporate open source governance, we took a practice-based approach looking for companies with an advanced understanding and experience in corporate open source governance. For the population of our qualitative survey, we searched for companies that used open source software in their products or when providing services. Following our method of a qualitative survey [77], we aimed at collecting deep insights from each of the companies. Therefore, our research relied on theoretical sampling with a small, purposive, and non-probabilistic sample of companies we had direct access to. We aimed at choosing a small sample, with which we could work on a deep and detailed level and conduct multiple semi-structured interviews with open source governance experts in each of these companies. We did recognize the limitations of a small sample, but decided that for an exploratory study on such a novel topic we would benefit more from depth rather than breadth in our sampling. We also recognized that in such a sample, a single observation was sufficient for inclusion in the QDA coding system [112] [130].

We started by defining a set of sampling criteria we would use to categorize the companies in our research group's industry network of about 140. We defined the following criteria or sampling dimensions:

- By type of business model
- By type of customer
- By market position
- By size
- By maturity.

The type of business model dimension categorized companies by how they made money, in terms of the products they sold or services provided. The type of customer dimension categorized companies by their customers and markets. The market position dimension categorized companies by their market share. The size dimension categorized companies by their number of employees or market capitalization. The maturity dimension categorized companies by their growth.

We then defined different assessment options for each of the criteria to apply to our network of companies:

- By type of business model
  - Software product companies
    - Closed/ proprietary
    - Open source business model (single-vendor or distributor)
    - Other products incorporating software
  - Services firms
    - Software development services
    - Governance tool providers
    - Management consulting
  - Non-profits
    - Open source foundations
    - Standards bodies
- By type of customer
  - Enterprise customers
  - Retail customers
  - Government
- By market position
  - Monopolist
  - Leader
  - Also running
  - Laggard
- By size

  - Monopolist
  - Leader
  - Also running
  - Laggard

- By maturity

  - Mature
  - Growth
  - Startup.

We went through all the companies in our network, assessing them using the above-mentioned dimensions. This resulted in a table of companies, dimensions, and their sampling assessment. For an excerpt from the full theoretical sampling, see Figure 3.2.

Dimensions		By type of organization (what they make their money off)								By type of customer			By market position				By size (employees) / market capitalization			By maturity		
		Software product companies				Services firms			Non-profits													
Companies	Editor	Closed/ proprietary	Open source business model (single-vendor or distributor)	Other products incorporating software	Software development services	Governance tool providers	Management consulting	Open source foundations	Standards bodies	Enterprise customers	Retail customers	Government	Monopolist	Leader	Also running	Laggard	Large	Medium	Small	Mature	Growth	Startup
ABB	DR	x		x	x					x				x			x			x		
Actano	DR	x								x					x				x	x		
Alfresco	DR		x							x				x				x			x	
Andreria	DR				x					x		x		x				x			x	
Apache Software Foundation	NH							x		x	x			x					x	x		
BG Phoenics	DR	x								x		x			x			x		x		
Bund - Umweltbundesamt	DR	x										x	x					x		x		
Bund - Wasser	DR	x										x	x					x		x		
Bundesdruckerei	DR	x								x		x		x				x		x		
Canoo	DR				x					x					x			x		x		
Ciber	DR				x					x					x			x		x		
Cosmocode	DR		x		x					x					x				x	x		
Creativ	DR		x		x					x				x				x			x	
DB Systel	DR	x		x	x					x					x		x			x		
DLZ IT	DR	x							x		x	x	x					x		x		
Docufy (DE)	DR	x								x				x					x	x		
Eclipse Foundation	NH							x		x	x			x					x	x		
Evidanza	DR				x					x					x			x		x		
Exasol	DR	x								x				x				x			x	
HIS	DR	x	x						x			x	x	x				x		x		
IT-Agile	DR				x					x				x					x			
IVU	DR	x								x				x				x		x		
KDAB	DR		x							x	x			x				x			x	
Kisters	DR	x								x					x			x		x		
Knowis	DR	x								x					x			x		x		
Linux Foundation	NH							x		x	x	x		x					x	x		
Main Donau Netz	DR			x					x	x			x	x				x		x		
Mathema	DR				x					x					x			x		x		
Mayflower	DR				x					x					x			x		x		
MicroDoc	DR	x			x					x					x			x		x		
Mister Spex	DR	x		x							x			x					x			x
Movepilot	DR	x		x						x	x			x						x	x	
New Store	DR	x								x	x			x				x			x	x
PSI	DR	x								x				x				x		x		
QAware	DR	x			x					x					x				x		x	
Quinscape	DR				x					x					x			x			x	
Schema	DR	x								x				x				x			x	
Sebamed	DR	x		x						x					x			x		x		
sepp.med GmbH	DR	x		x						x					x			x		x		
Semet	DR		x							x			x	x				x		x	x	
Silbury	DR				x					x					x			x		x	x	
Solyp	DR	x								x					x				x		x	x
SysEleven	DR		x		x					x				x	x			x	x		x	
Tavendo	DR		x							x	x	x		x					x			x
Accenture	NH	x		x		x	x			x		x	x			x	x			x		
Adidas	NH			x						x	x			x		x	x			x		
AGT International	NH	x								x		x			x		x			x		
Akamai	NH	x			x	x	x			x		x		x			x			x		
Alcatel-Lucent (aquired by Nokia in 2015)	NH	x			x					x					x		x			x		
Amazon	NH	x		x	x	x				x	x	x	x				x			x	x	
Astrum IT	NH	x		x		x				x						x			x		x	
Audi	NH	x		x							x				x		x			x		
AVL DITEST	NH	x		x	x					x				x			x			x		
BearingPoint	NH					x	x			x		x			x		x				x	
Black Duck Software	NH	x				x	x			x				x					x		x	
BMW Car IT	NH	x		x	x					x					x			x		x		
Bosch	NHI	x		x						x	x	x		x			x					
Brightone	NH	x			x					x					x			x			x	
Brose	NH			x						x				x			x			x		
BSHG	NH			x							x			x			x			x		
BVG	NH	x		x							x	x	x				x			x		
Capgemini	NH	x	x		x	x	x			x				x			x			x		
Clear IT	NH	x			x					x						x		x			x	
Consorsbank	NH			x							x				x			x		x		
DATEV	NH	x			x	x	x			x	x			x			x			x		
Deutsche Bahn	NH	x		x							x		x				x			x		
Develop-Group	NH	x			x		x			x					x				x		x	
Dräger	NH			x						x		x		x			x			x		

Figure 3.2: Theory Building – Excerpt from Theoretical Sampling

In our first sampling iteration, we chose and contacted nine companies in a way to ensure a polar (diverse as proposed by Jansen [77]) theoretical sample – companies with different combinations of sampling criteria assessments covering a broad set of companies with the common denominator being their use of open source and experience in corporate open source governance. As a result, we selected companies with profiles such as:

- A large American semiconductor and telecommunications equipment company operating internationally in several hardware and software domains, with a business model of selling products incorporating software (including open source) to business customers, with a leading position and large market share, and having reached its maturity.
- A medium-sized German company operating in Europe in the enterprise software domain, with a business model of being a software product vendor for open source software, with an also runner market position, and still growing.
- A small Canadian organization operating internationally as an open source foundation, with a business model of being a non-profit foundation creating, maintaining, and guiding open source projects and communities, with a leading position among similar foundations, and having reached its maturity.

After contacting the first nine companies chosen in the first sampling iteration, we got replies from some and scheduled the first expert interviews. Overall, we interviewed five experts at four companies as a result of the first sampling iteration. We then selected more companies from our sampling spreadsheet to further ensure that our sampling covered a broad set of companies with experience in FLOSS governance. This led us to the second sampling iteration, during which we sent reminders to the idle companies from the first iteration, while also selecting and contacting seven new companies with similar profiles to the latter. As a result of the second sampling iteration, we interviewed six experts from four companies. After the first two iterations we already started analyzing the data, during which we identified what aspects of corporate open source governance were covered in a lesser detail (for example legal aspects of governance in terms of license compliance in outbound

compliance and legal aspects of supply chain management governance), which shaped our next sampling iteration. In the third sampling iteration, we contacted four more companies, which resulted in two more expert interviews at two companies. In the last, fourth sampling iteration we contacted nine more companies to fill in any final gaps our theory would potentially have (based on our ongoing data analysis). This resulted in eight final expert interviews at five new companies.

We started the theoretical sampling iterations in November 2016 (with the first interviews taking place in February 2017) and conducted the last iteration in September 2018 (with the last interviews taking place in October 2018). As a result, overall we chose 15 companies (interviewing 21 experts) sampled from our industry network of about 140 companies with advanced FLOSS governance practices. We conducted polar theoretical sampling to cover a diverse and representative set of companies, which resulted in a sample with highly varying characteristics [45] [77]. The list of companies and some of their sampling characteristics are presented in Table 3.1. Company names were anonymized per their request.

After a company was selected and contacted, we identified the potential experts of open source governance within the company with access to important information. When possible, we aimed at interviewing more than one expert from a company for data triangulation (as a theoretical instrument to increase the internal validity of our study). We aimed to have a wide range of types of interviewees who have experienced the circumstances relevant to our research topic [109]. In particular, when possible, we talked to open source program officers (coordinators), managers involved in corporate open source governance, and software developers using open source software components and tools.

Company	Company Domain	By Size	By Type of Customer	By Business Model
Company 1	Enterprise Software	Medium	Enterprise, Retail	SP-OS, MC, GT
Company 2	Automotive	Medium	Enterprise	SDS
Company 3	Consulting	Medium	Enterprise	SP-OS, SDS
Company 4	FLOSS Foundation	Small	Enterprise, Retail	OSF
Company 5	Enterprise Software	Medium	Enterprise, Retail	SP-OS
Company 6	Automotive	Medium	Enterprise	SDS
Company 7	Enterprise Software	Medium	Enterprise, Retail	SP-CS
Company 8	Hardware and Software	Large	Enterprise, Retail, Government	SP-OS, SP-CS, OP, GT
Company 9	Legal	Large	Enterprise, Government	MC
Company 10	Hardware and Software	Large	Enterprise	OP
Company 11	Enterprise Software	Medium	Enterprise	SP-OS
Company 12	Consulting, Enterprise Software	Large	Enterprise	MC, SDS
Company 13	Enterprise Software	Small	Enterprise	SP-CS, SDS, GT
Company 14	Enterprise Software	Medium	Enterprise	SDS
Company 15	Enterprise Software	Large	Enterprise, Retail	SP-OS, SP-CS, MC, GT

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.

Table 3.1: Theory Building – Sampled Companies

Company 1 was a German company operating internationally in the enterprise software domain. It was a medium-sized company selling software in both B2B and B2C markets (though the main

focus was on enterprise customers). It had an open source business model being a distributor of a widely used open source software, which explained the company's deep understanding and expertise in corporate open source governance. The company was also a governance tool provider and provided consulting services. It was not the market leader, but had a sizable market share. The company reached its maturity and was not growing anymore.

Company 2 was a Polish company operating mainly in Europe in the automotive domain. It was a medium-sized company providing software development services to enterprise customers. It used open source tools and libraries in proprietary software development, which explained the company's expertise in corporate open source governance. The company was not the market leader, but had a sizable market share. It did not reach its maturity and was still growing (being recently acquired by a larger company).

Company 3 was a German company operating internationally in the consulting domain. It was a medium-sized company providing open source consulting services and tools to enterprise customers. It had an open source business model providing open source specific consulting, distributing open source software, and providing software development services (also using open source). The company was heavily focused on open source software and support, which explained the company's expertise in corporate open source governance. The company was a market leader and had a sizable market share. It did not reach its maturity and was still growing.

Company 4 was a Canadian foundation operating internationally as an open source foundation. It was a small-sized foundation creating, maintaining, and guiding open source projects and communities, which explained the organization's expertise in corporate open source governance. It worked with both individual developers, and with partner companies. It one of the leading and mature open source foundations worldwide.

Company 5 was an Italian company operating internationally in the enterprise software domain. It was a medium-sized company with a business model of a software product vendor for open source software, which explained the company's expertise in corporate open source governance. It operated

in both B2B and B2C markets (though the main focus was on enterprise customers). The company was a market leader and had a sizable market share. It did not reach its maturity and was still growing.

Company 6 was a medium-sized subsidiary of a large German company operating internationally in the automotive domain. Its business model as a subsidiary was in being an internal software supplier and providing software development services to the main (owner) company. It was an active user of open source software and was involved in a number of open source initiatives (some started and cosponsored by the company), which explained the company's deep understanding and expertise in corporate open source governance. As a subsidiary of another company, it competed with other suppliers to the same company, among which it was one of the main suppliers. The company reached its maturity and was not growing anymore.

Company 7 was a German company operating in Europe in the enterprise software domain. It was a medium-sized company selling software in both B2B and B2C markets. It had an open source business model being a software product vendor for open source software, which explained the company's deep understanding and expertise in corporate open source governance. The company was a market leader with a large market share. The company did not reach its maturity and was still growing.

Company 8 was an American multinational corporation and technology company operating internationally in several hardware and software domains. It was a large company selling products in both B2B and B2C markets, as well as to government customers. In its different parts, it had different business models, including being a software product vendor for open source software and for closed source software, providing governance tooling, and selling other (hardware) products incorporating software. It was a leading company in terms of open source use, contribution, and corporate open sourcing, which explained the company's deep understanding and expertise in corporate open source governance. The company was a market leader with a large market share in several markets. The company reached its maturity and was not growing anymore.

Company 9 was an American multinational law company operating internationally in several legal domains. It provided legal services and management consulting to business customers and government customers on various legal issues related to software. The company had experts specialized in the legal aspects of open source software, which explained the company's deep understanding and expertise in corporate open source governance. It was a market leader with a large market share. The company reached its maturity and was not growing anymore.

Company 10 was an American multinational semiconductor and telecommunications equipment company operating internationally in several hardware and software domains (though mostly focused on hardware). It was a large company selling products in B2B markets. The business model was based on hardware products incorporating software, including open source software and being a leader in a number of open source initiatives and communities, which explained the company's deep understanding and expertise in corporate open source governance. The company was a market leader with a large market share in several markets. The company reached its maturity and was not growing anymore.

Company 11 was a German company operating in Europe in the enterprise software domain. It was a medium-sized company with a business model of a software product vendor for open source software, which explained the company's expertise in corporate open source governance. It operated in a B2B market. The company was not the market leader, but had a sizable market share. It did not reach its maturity and was still growing.

Company 12 was a German company operating internationally in the enterprise software and consulting domains. It had a business model based on providing software development services and management consulting to business customers. Among other topics, it consulted companies on open source use, governance, and compliance, which explained the company's deep understanding and expertise in corporate open source governance. The company was not the market leader, but had a sizable market share. It did not reach its maturity and was still growing.

Company 13 was a German company operating in Europe in the enterprise software domain. It

was a small company with business models of being a software product vendor for closed source software, providing software development services and governance tools. It operated in a B2B market. The company was using open source software in its products, which explained the company's deep understanding and expertise in corporate open source governance. The company had a laggard position in the market only with a small market share. It did not reach its maturity and was still growing.

Company 14 was a German company operating in Europe in the enterprise software domain. It was a medium-sized company with a business model of providing software development services to business customers. The company was using open source tools in software development, which explained the company's deep understanding and expertise in corporate open source governance. The company was not the market leader, but had a sizable market share. It did not reach its maturity and was still growing.

Company 15 was a German company operating internationally in the enterprise software domain. It was one of the largest companies in its domain. The company followed business models of selling closed source software products and governance tool, as well as providing management consulting. It operated in both B2B and B2C markets (though the main focus was on enterprise customers). It was a heavy user of open source tools and components with experience in open source compliance (engaging in working groups and initiatives around open source use and compliance), which explained the company's deep understanding and expertise in corporate open source governance. The company was a market leader with a large market share in several markets. The company reached its maturity and was not growing anymore.

3.3.2 Data Gathering

To collect data for our qualitative survey, we conducted semi-structured interviews with open source governance experts working at the companies in our sample. Our primary contacts at each of the companies introduced us FLOSS governance experts within their companies. Such experts had dif-

ferent roles, including but not limited to:

- Engineering Manager
- Project Manager
- Lawyer
- Head Business Structure, Systems and IT
- Open Source Compliance Manager, etc.

Expert employees covered different aspects of corporate open source governance. A diverse set of experts enabled us to answer our broad set of research questions. As presented in the section on theoretical sampling, we went through four round of sampling iteration. After each iteration, analyzing the gathered data we identified the potential gaps in our theory. In every subsequent sampling iteration, we looked for expert interviewees that could provide data on the lacking aspects (gaps) of the corporate open source governance. For example, after the first two sampling iterations and preliminary data analysis, we recognized that we had limited data on the legal aspects of open source governance (e.g. license compliance, license interpretation, etc.), which prompted our search of experts that could address the issue in the third sampling iteration. As a result, we interviewed a lawyer from Company 9 (Interview CX9.1 in Table 3.2) and the vice president and legal counsel from Company 10 (Interview CX10.1 in Table 3.2).

Figure 3.3 illustrates the geographic distribution of the experts we interviewed for theory building with most experts comping from Germany – 14 experts, and the rest from the USA, Canada, France, Italy and Poland. The countries interviewees were based in are color-coded based on the number of interviewees per country.

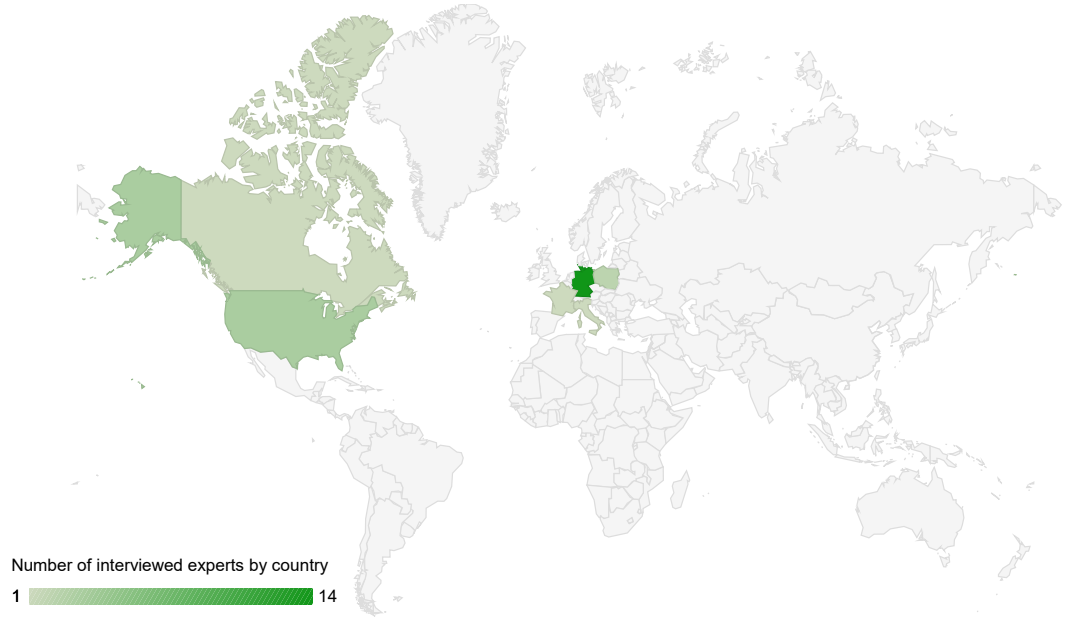


Figure 3.3: Theory Building – Map of Interviewee Countries

Using semi-structured interviews as our main survey instrument, we conducted the interviews in an iterative manner adjusting the questions after each iteration, yet keeping the core topics of the questions intact. Find the last iteration of the interview questions for theory building in Section C.1 in Appendix C.

In addition to the expert interviews (our main data source), we also collected primary materials on corporate open source governance, including:

- white papers
- company guidelines
- slides and practitioner reports.

For example, we studied Google's company-internal documentation / guideline on FLOSS use governance[2], which was publicly released in March 2017. Table 3.2 presents the full overview of the main data sources we used in theory building – the expert interviews at the sampled companies we conducted and used in the qualitative survey.

[2]Google's Guideline on FLOSS Use Governance – `https://opensource.google.com/docs/using/`

Interviewee	Company	Interviewee Role	Interview Date	Sampling Iteration
CX1.1	Company 1	CTO, Director of Open Source	2017-02-24	First
CX2.1	Company 2	Engineering Manager, Project Manager	2017-03-02	First
CX2.2	Company 2	Engineering Manager, Project Manager	2017-03-02	First
CX3.1	Company 3	CEO	2017-03-08	First
CX4.1	Company 4	Director of Open Source Projects	2017-07-07	First
CX5.1	Company 5	Head of Policy and Innovation	2017-03-30	Second
CX6.1	Company 6	Open Source Compliance Manager	2017-04-12	Second
CX7.1	Company 7	Engineering Manager	2017-07-21	Second
CX7.2	Company 7	Engineering Manager	2017-07-21	Second
CX7.3	Company 7	Head Business Structure Systems and IT	2017-07-21	Second
CX8.1	Company 8	Senior Open Source Compliance Engineer	2018-02-16	Second
CX9.1	Company 9	Lawyer	2017-07-17	Third
CX10.1	Company 10	Vice President, Legal Counsel	2017-07-18	Third
CX11.1	Company 11	VP Strategy	2018-01-24	Fourth
CX12.1	Company 12	IT Consultant	2018-01-24	Fourth
CX13.1	Company 13	Head of Service and Support	2018-02-28	Fourth
CX14.1	Company 14	Engineering Manager	2018-04-12	Fourth
CX14.2	Company 14	Senior Legal Counsel	2018-04-12	Fourth
CX14.3	Company 14	Legal Compliance Reviewer	2018-04-12	Fourth
CX15.1	Company 15	Director of Open Source	2018-10-28	Fourth
CX15.2	Company 15	Open Source Compliance and Audit Manager	2018-10-28	Fourth

Table 3.2: Theory Building Data Sources – Expert Interviews

Table 3.3 presents the rest of the data sources we used in theory building – the primary materials from companies with an advanced understanding of open source governance including their white papers, slides, and guidelines.

ID	Primary Material	Type	Organization	Date
PM1.1	Best Practices in Open Source Governance	White Paper	Hewlett-Packard	2007-09-26
PM2.1	Linux Foundation Compliance Program: Generic FOSS Policy	Guideline	The Linux Foundation	2012-04-22
PM2.2	License Scanning and Compliance Programs for FOSS Projects	Guideline	The Linux Foundation	2018-02-26
PM3.1	Open Source Software and Patents: How the GPLv3 Affects Patent Portfolios	White Paper	Osborne Clarke	2013-02-05
PM4.1	A Smart Way to Manage OSS Compliance with Yocto+SPDX	Slides	Fujitsu	2016-07-13
PM5.1	The Corporate Counsel's Guide to OSS Policy Implementation	Guideline	Black Duck Software	2016-09-08
PM5.2	Open Source Software Risk Maturity Model	White Paper	Black Duck Software	2016-11-30
PM5.3	2017 Open Source Security and Risk Analysis	White Paper	Black Duck Software	2017-04-18
PM6.1	Internal Documentation on Open Source Use Governance	Guideline	Google	2017-03-29
PM6.2	Open Source Casebook: Authorship in Open Source	Guideline	Google	2017-03-29

PM6.3	Open Source Casebook: Contract and Copyright Remedies	Guideline	Google	2017-03-29
PM6.4	Open Source Casebook: Trademarks in Open Source	Guideline	Google	2017-03-29
PM6.5	Internal Documentation on Open Source Governance Tools	Guideline	Google	2017-03-29
PM7.1	Using Open Source Code	Guideline	TODO Group	2017-09-12
PM8.1	Implementing and Managing Open Source Compliance Programs	Slides	Samsung	2017-11-17
PM9.1	Reuse Best Practices	Guideline	Free Software Foundation Europe	2017-12-14
PM10.1	Open Source at Scale	Slides	Microsoft	2018-12-07
PM11.1	Report of Studies into Business Workflows and Defined Roles for Software Development	White Paper	OpenChain Project	2018-12-12
PM11.2	The OpenChain Open Source Policy Template Draft	Guideline	OpenChain Project	2018-12-14
PM12.1	The Tidelift Guide to Managing Open Source	Guideline	Tidelift	2019-01-28

Table 3.3: Theory Building Data Sources – Primary Materials

3.3.3 Data Analysis

We analyzed the gathered qualitative data from expert interviews and primary materials following the qualitative survey research method [77]. The analysis of the primary materials helped us increase

the internal validity of our survey through data triangulation. Data analysis was split into three levels / phases:

- 1st-level analysis – unidimensional description
- 2nd-level analysis – multidimensional description
- 3rd-level analysis – explanation.

In the first phase, we conducted open coding. We created a basic set of labels for the core concepts of corporate open source governance that emerged from the initial analysis of the gathered data. We documented these codes. We created an unsorted list of labels (codes) and assigned each label to one or more text segments (codings). As to the coding granularity, we coded both sentences and paragraphs. This resulted in the preliminary codebook – a spreadsheet with the key concepts of FLOSS governance that came up repeatedly during the interview and primary materials analysis. Open codes were direct annotations of the interviews and primary materials, ensuring the early links of the emerging theory to the data traces.

In the second phase, we conducted axial coding. We used the codes and codings from the first phase of data analysis to build a hierarchical structure of the developed codes from the codebook. We grouped them into categories to form a map of concepts supported by the analyzed data. Each category represented a core aspect of corporate open source governance. Each broad category (e.g. inbound governance, supply chain management) became a top-level code category, which other lower-level codes in the hierarchy that would detail the concepts of governance. The lowest level codes in each code hierarchy corresponded to specific industry best practices identified during the data analysis of the expert interviews and primary materials. We used these codes not only to capture the proposed best practices (solutions) to given problems from one or more data sources, but also to address the problem and the context related to the proposed solution. We also defined the types of relationships between different best practices (e.g. sequential, hierarchical) that translated into the proposed process templates in our theory. As a result of this phase, we modified the codes as needed

to eliminate the unused codes, merging the codes that addressed similar concepts, splitting concepts that encompassed too many subtopics (e.g. multiple best practices), and modified codes when they were unclear or imprecise. We ended up with a codebook and a code system that captured the key concepts of corporate open source governance, which became the backbone of our theory.

In the third phase, we conducted selective coding. In this phase, the codebook did not change anymore, but we extended the code system by adding new coded segments (coding) from the gathered data. No more high-level codes were used, but the low-level codes were applied once again to all the relevant text segments from our qualitative data. These codings supported the outlined concepts of open source governance providing specific best practices and examples. We then used the resulting qualitative data analysis to answer the research questions we had asked. We used the data to present the broad categories of FLOSS governance, their subtopics, and specific best practices. We explained each of the proposed best practices presenting their contexts, problems, and solutions, as well as their relationships to other best practices.

During data analysis, we iteratively extended and modified the code system to include all the key concepts of corporate open source governance emerging from the data. We iteratively went through each of the above-mentioned phases. After each QDA iteration, we assessed if more interviews are needed to further analyze and understand the identified concepts, or if there is a potential to find not yet identified ones, which was followed by a new round of sampling. See Table 3.2 for the overview of the expert interviews we conducted in each sampling iteration. After the fourth sampling iteration and the subsequent data analysis, we could only identify a few minor subtopics (subcodes) of open source governance, the code system was not significantly modified. Therefore, we considered achieving theoretical saturation, which indicated that we had enough data for comprehensive theory building as no more interviews were needed to identify new concepts or to better understand the ones that were already captured in the code system [62] [48] [119].

See the resulting code system with the number of codings per code in Section D.2 in Appendix D.

3.3.4 Theory Presentation

To present our theory in a structured, understandable, and applicable manner, we opted for developing a handbook of corporate open source governance that consisted of industry best practice patterns and their interconnections as proposed process templates / workflows.

Patterns and pattern languages have been used in the past to present different concepts of open source use, development, and governance. Among others, Hannebauer and Gruhn [64] presented an overview of the current state of research on OSS patterns, including 40 published patterns, their key topics, and relationships between them. Some of these patterns focused on open source development [65] [66] and contributions to OSS communities [156] [67], while some focused on the commercial use of open source (broadly related to our research) [98] [150] [151].

In our theory patterns corresponded to individual best practices. Each best practice used the same structure, including:

- ID
- Name
- Actor
- Context
- Problem
- Solution.

At their core, these best practices are Context-Problem-Solution triplets. Topically related best practice patterns – mirroring the qualitative data analysis – formed the subsections and sections focused on different aspects of corporate open source governance. The top-level sections corresponded to the core best practice categories, such as inbound governance, outbound governance, and supply chain management. Each answered one research question we had asked. Together these top-level sections, their subsections, and individual best practices made up the handbook of corporate open source governance.

The best practices within subsections were linked to each other through in-text references of other best practice patterns. Such links resulted from our data analysis and could be of different types depending on the relationship a link described. For example if sequential links denoted that a given best practice would come after or before another one. Hierarchical links showed top-level best practices referring to more specific patterns. Alternative links showed two or more best practices that could be applied in a certain situation / context. In our handbook we used italic font and cross-references for the inter-BP links. An example cross-reference link would be "→ *establishing a board of stakeholders to organize the transition*" from the best practice A.1.2 (OSGOV-GETSTA-TRAORG-2. Designate the transition manager) presented in its context below:

"After → *establishing a board of stakeholders to organize the transition* towards regulated FLOSS governance at the company, you need to delegate the management of the transition process to a responsible person."

Each section had a descriptive name that captured the domain of best practices it collected. The section hierarchy represented a hierarchical breakdown of the overall domain of FLOSS governance. The lowest-level (leaf) nodes corresponded to best practice descriptions. A single best practice pattern abstracted from a single example, representing a general proposition that could be widely applicable within its context.

The subsections of the handbook also presented optional workflows that linked different best practices within the subsection. These workflows or governance process templates emerged from our data analysis and would be relevant to the practitioners using our handbook as they could modify and apply the workflows at their companies.

Figure 3.4 illustrates the common pattern structure we used to present our theory.

Short	Name	
Main responsible	**Actor**	
(Abstract form of)	**Context**	of problem
(Abstract form of)	**Problem**	at hand
(Abstract form of)	**Solution**	of problem

Figure 3.4: Theory Presentation – Best Practice Pattern

See an example of a best practice (BP) pattern in Table A.1.1 in Appendix A. See an example of a process template (PT) or workflow in Figure A.1 in Appendix A. See an example of a top-level handbook category with a hierarchy of governance concepts and interconnected best practices in Appendix A.

3.4 Industry Best Practices for Corporate Open Source Governance

In this dissertation we propose a theory of industry best practices for corporate open source governance based on our qualitative survey of industry experts and primary materials. Our theory answered our four research questions and addressed each of them. Addressing each of the broad research questions and given the breadth of the topic of FLOSS governance, we limited the scope of this three-year research project to cover a mix of basic, intermediary and advanced subtopics, including:

- Getting Started with Open Source Governance
  - Product Analysis
  - Transition Organization
  - Transition Policy
  - IP-at-Risk Analysis
  - Communication and Capabilities
- Inbound Governance
  - Component Search
  - Component Selection
  - Component Approval
  - Component Repository and Reuse
  - Component Monitoring
  - Engineering Management
  - Communication
  - Education
- Supplier Management
  - Supply Chain Management Policy

  - Supply Chain Management Process
  - Preventive Governance
  - Corrective Governance
  - Bill of Materials Management
  - License Compliance for Supply Chain

- Outbound Governance

  - License compliance
  - Distribution Preparation
  - Product Distribution
  - Release Management

- General Governance

  - Governance Management
  - Open Source Program Office
  - License Interpretation
  - Capabilities.

The above-mentioned concepts of corporate open source governance emerged from our qualitative survey and constituted the core topics our theory addressed. We developed and presented a number of proposed industry best practices in full detail in the following categories of the theory:

- Getting Started with Open Source Governance

  - Product Analysis (OSGOV-GETSTA-PROANA) - 8 best practices
  - Transition Organization (OSGOV-GETSTA-TRAORG) - 8 best practices
  - Transition Policy (OSGOV-GETSTA-TRAPOL) - 3 best practices
  - IP-at-Risk Analysis (OSGOV-GETSTA-IPRISK) - 9 best practices
  - Communication and Capabilities (OSGOV-GETSTA-COMCAP) - 5 best practices

- Inbound Governance (OSGOV-INBGOV)

  - Component Approval (OSGOV-INBGOV-COMAPP) - 13 best practices
  - Component Reuse (OSGOV-INBGOV-COMREU) - 19 best practices
- Supply Chain Management (OSGOV-SUCHMA)
  - Supply Chain Management Policy (OSGOV-SUCHMA-SCMPOL) - 3 best practices
  - Supply Chain Management Process (OSGOV-SUCHMA-SCMPRO) - 5 best practices
  - Preventive Governance (OSGOV-SUCHMA-PREGOV) - 4 best practices
  - Corrective Governance (OSGOV-SUCHMA-CORGOV) - 4 best practices
  - Bill of Materials Management (OSGOV-SUCHMA-BOMMAN) - 4 best practices
  - License Compliance for Supply Chain (OSGOV-SUCHMA-LICCOM) - 2 best practices
- Outbound Governance (OSGOV-OUTGOV)
- General Governance (OSGOV-GENGOV).

We also identified and presented industry best practices on General Governance and Outbound Governance, but did not present them in the Context-Problem-Solution pattern format. We had to prioritize and choose parts of our theory that we could present in this actionable format that required significant time to apply to the proposed theory.

We present our theory's take on getting started with open source governance in Section 3.4.1. We present our theory's take on inbound open source governance in Section 3.4.2. We present our theory's take on supply chain management governance in Section 3.4.3. We present our theory's take on outbound open source governance in Section 3.4.4. We present our theory's take on general open source governance in Section 3.4.5.

3.4.1 Getting Started

Key Theory Topic Overview – Getting Started

- Product Analysis (OSGOV-GETSTA-PROANA) - 8 best practices
- Transition Organization (OSGOV-GETSTA-TRAORG) - 8 best practices
- Transition Policy (OSGOV-GETSTA-TRAPOL) - 3 best practices
- IP-at-Risk Analysis (OSGOV-GETSTA-IPRISK) - 9 best practices
- Communication and Capabilities (OSGOV-GETSTA-COMCAP) - 5 best practices

Answering the research question *RQ-TB1*, we found that companies getting started with open source governance should start with a transition organization guided by a transition policy. The transition policy helps a company define its principles in regard to open source use and governance. The transition organization then operationalizes the principles defined in the policy, turning them into a process that involves different stakeholders that have been or would be using open source components in products, or making decisions regarding open source governance. The transition organization starts with establishing a board of stakeholders that oversees and organizes the transition that includes defining the transition timeline and scope, as well as implements the transition process.

In this section, we discuss the subtopics and specific best practices for getting started with open source governance. The subsection on Product Analysis presents the insights on the product analysis subtopic from our theory's take on getting started with corporate open source governance. The subsection on Transition Policy presents the insights on the transition policy aspect when getting started with corporate open source governance. The subsection on Transition Organization presents the insights on the transition organization aspect when getting started with corporate open source governance. The subsection on IP-at-Risk Analysis presents the insights on the IP-at-risk analysis aspect when getting started with corporate open source governance. The subsection on Communication and Capabilities presents the insights on the communication and capabilities aspect when getting started with corporate open source governance.

Product Analysis

Our theory summarizes a number of industry best practices on the scanning of the software product code for license compliance, creating a product architecture model including open source components and their metadata, and running and documenting open source use analysis in products. Best practices in this category include:

- OSGOV-GETSTA-PROANA-1. Use a combination of methods for product analysis
  - OSGOV-GETSTA-PROANA-1.1. Use one mandatory survey for initial assessment
  - OSGOV-GETSTA-PROANA-1.2. Establish a process of continuous reporting and assessment
  - OSGOV-GETSTA-PROANA-1.3. Select and use governance tools for automation
- OSGOV-GETSTA-PROANA-2. Establish and use a product architecture model
  - OSGOV-GETSTA-PROANA-2.1. Create product architecture model
  - OSGOV-GETSTA-PROANA-2.2. Maintain product architecture model
- OSGOV-GETSTA-PROANA-3. Run use analysis
  - OSGOV-GETSTA-PROANA-3.1. Run open source use analysis in products
  - OSGOV-GETSTA-PROANA-3.2. Document current open source use.

Product analysis is a critical part of getting started with open source software. Before setting up open source governance processes, a company must identify and analyze the current use of open source components that have been used but not approved or documented before. Proposed best practices in this category describe methods for analyzing the current use of open source including a mandatory survey for initial situation assessment, and ways to document the identified open source component and their metadata. After the initial assessment, companies should establish a process of continuous reporting and assessment for the open source components used from that point on, which is described in detail in an example best practice from our theory in Table 3.4.

Another industry best practice is the creation of a product architecture model to set up and maintain a structured and formalized view of software components used. Companies should define open source component-specific properties within the model to allow collection, tracking, maintenance, and monitoring of metadata including open source license information, export restrictions, known security vulnerabilities, and software dependencies. If possible, the product architecture model should be integrated into the build process or continuous development process to ensure higher automation.

Here is an example of a best practice from this subsection:

Name	OSGOV-PROANA-1.2. 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 licence 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.

Table 3.4: Best Practice OSGOV-PROANA-1.2. Establish a process of continuous reporting and assessment

The industry best practices of our theory can be traced to the data from the qualitative survey we performed. Here is an example of such a trace from Company 14's legal counsel responsible for open source compliance talking about the specifics of establishing a process of continuous reporting and assessment of open source components:

> *"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." —*CX14.2*

Here is another example of such a trace from Company 6's open source compliance manager:

> *"We ask the developers to report those kinds of components that have this kind of licenses, and then the license checks the components and the rough context is documented and the system goes to a board, company-wide board where we have software developers, the compliance managers, the lawyers and a patent lawyer, a copyright lawyer. A group sits together and then discusses that and makes decisions on the license terms."* —*CX6.1*

The industry best practices in our theory are interconnected forming workflows or process templates that practitioners can use to apply our handbook for corporate open source governance. We present an example process template from this subsection in Figure 3.5. This process template connects a subset of product analysis best practices. According to our theory, companies should start by using a combination of methods for product analysis including:

- mandatory survey for initial assessment of the previously used open source components
- automated ongoing assessment of the previously used and new added (during the transition) open source components.

At the same time, industry experts recommend establishing a process of continuous reporting and assessment of the newly added open source components during the transition process. After the transition is over, companies should establish long-term inbound governance processes for continuous governance. Next, companies need to analyze the identified open source software in products and document the current FLOSS use.

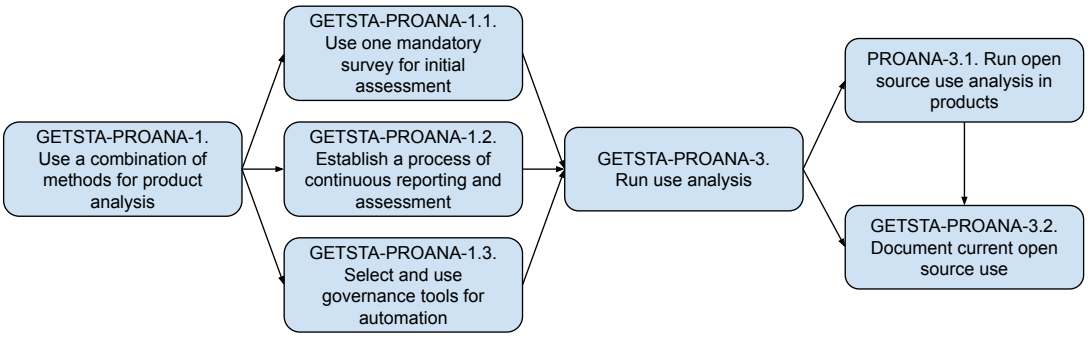


Figure 3.5: Example Process Template – Product Analysis

See Section A.3 in Appendix A for the process templates and for the complete handbook subsection with detailed best practices on product analysis.

Transition Policy

We found that most companies establish guidelines for getting started with open source governance. We call these guidelines a transition policy, which must be established, communicated, and continuously adjusted and improved. The transition policy outlines the principles for the transition, but does not cover any operational aspects of the transition, which is done through the transition organization. Transition policy related practices include:

- OSGOV-GETSTA-TRAPOL-1. Establish FLOSS governance policy for the transition period
- OSGOV-GETSTA-TRAPOL-2. Communicate FLOSS governance policy for the transition period
- OSGOV-GETSTA-TRAPOL-3. Adjust and improve FLOSS governance policy for the transition period

FLOSS governance policy for the transition period covers all the critical issues around the use of open source components in products, such as license compliance, bill of materials management, documentation, and communication. The policy can be stored as a single document or divided into two separate documents, as described in Table 3.5.

Here is an example of a best practice from this subsection:

Name	OSGOV-GETSTA-TRAPOL-1. Establish FLOSS governance policy for the transition period
Actor	Transition manager, Transition board
Context	Developers have been using open source components without any governance guidelines in the past. In parallel to introducing FLOSS governance at the company, you need to define and communicate the new rules for using open source components for the transition period, before → *implementing the transition process*.

Problem	How should you manage FLOSS governance during the transition period, before you have established a comprehensive FLOSS governance strategy?
Solution	In parallel to → *establishing the transition process*, establish FLOSS governance policy for the transition period that covers all the critical issues around the use of open source components in products, such as license compliance, bill of materials management, documentation, communication, etc. The governance policy can be stored as a single document or divided into two separate documents: • The second establishes a set of standards and tasks for the employees to follow to ensure compliance with FLOSS governance processes. This way, the policy can be implemented across the whole company under identical conditions. Also at larger companies, each division or department can adopt the policy with certain differences. • The second establishes a set of standards and tasks for the employees to follow to ensure compliance with FLOSS governance processes. This way, the policy can be implemented across the whole company under identical conditions. Also at larger companies, each division or department can adopt the policy with certain differences. It is necessary to → *communicate FLOSS governance policy for the transition period* and to → *adjust and improve FLOSS governance policy for the transition period*.

Table 3.5: Best Practice OSGOV-GETSTA-TRAPOL-1. Establish FLOSS governance policy for the transition period

See Section A.2 in Appendix A for the process template and for the complete handbook subsection with detailed best practices on transition policy.

Transition Organization

We identified that a common pattern for organizing the transition to governance follows these industry best practices:

- OSGOV-GETSTA-TRAORG-1. Establish a board of stakeholders to organize the transition
- OSGOV-GETSTA-TRAORG-2. Designate the transition manager
- OSGOV-GETSTA-TRAORG-3. Define the responsibilities and tasks of the transition manager
- OSGOV-GETSTA-TRAORG-4. Start small, then replicate - define the scope of the transition process
- OSGOV-GETSTA-TRAORG-5. Define the transition timeline
- OSGOV-GETSTA-TRAORG-6. Establish the transition process
- OSGOV-GETSTA-TRAORG-7. Communicate the transition process
- OSGOV-GETSTA-TRAORG-8. Implement the transition process

We found that establishing a board of stakeholders to organize the transition is a starting point for getting started with open source governance. These stakeholders include the everyday users and decision makers in regard of open source, including but not limited to senior developers (known internally for their open source use and competency), engineering managers (usually de facto decision makers on FLOSS matters), lawyer (responsible for FLOSS license clearance and related issues), business/product managers, software architect, software procurement officer.

As to the transition process, we found that common aspects of such a process include:

- outlining the motivation behind FLOSS governance
- clarifying the roles of the employees during the transition
- communicating the timeline and scope of the transition

- communicating the steps of the transition (product analysis and risk mitigation) and expected outcomes
- setting up new and structured procedures for decision making related to governance.

We present the full best practice on establishing a transition process in Table 3.6.

Here is an example of a best practice from this subsection:

Name	OSGOV-GETSTA-TRAORG-6. Establish the transition process
Actor	Transition manager, Transition board
Context	After → *defining the scope of the transition process* and → *defining the transition timeline*, the transition manager and the board must establish the transition process that is first implemented at a pilot project in the company, then replicated in other parts of the company.
Problem	The transition process is often the determinant of the transition's success. The specifics of the transition process depend on the company, on its use cases in terms of FLOSS use and on the capabilities of the employees. What are some common principles for establishing a smooth transition process?
Solution	The transition process outlines the company's shift from unstructured, ad-hoc FLOSS governance to the structured and well-defined one. This handbook section covers how the company should get started with FLOSS governance. However, before the actual governance process, the pilot project and its team (and other projects consequently) must be exposed to the transition setup to prepare for the upcoming changes. Such a transition also ensures that the new governance approach and its motivation is clear to all the employees. During the transition, the transition manager and board are established and introduced to the employees, which is the basis for the implementation of all the other best practices.

The transition process should follow these principles:

- be clearly defined
- be easy to follow without constant guidance by the board
- be scalable and replicable
- be assisted with tools that automate and/or ensure compliance with the process.

The transition process should include:

- outlining the motivation behind FLOSS governance
- clarifying the roles of the employees during the transition
- communicating the timeline and scope of the transition
- communicating the steps of the transition (product analysis and risk mitigation best practices) and their expected outcomes
- setting up new and structured procedures for decision making related to governance.

Table 3.6: Best Practice OSGOV-GETSTA-TRAORG-6. Establish the transition process

The industry best practices in our theory are interconnected forming workflows or process templates that practitioners can use to apply our handbook for corporate open source governance. We present an example process template from this subsection in Figure 3.6. This process template starts by establishing a board of stakeholders to organize the transition and by defining responsibilities and tasks of the transition manager who coordinates the board. The board and the transition manager then establish the transition process, where they outlines the specific steps different employees need to undertake during the transition to FLOSS governance. Often companies do not transition to governance at once, but take an incremental approach. To achieve this, companies should follow

the best practice of starting small (e.g. from a pilot project), then replicating the small-scale transition in other parts of the company. Before establishing the transition process, companies also need to define the transition timeline taking into account the gradual nature of the transition and the specifics of the company at hand (e.g. how centralized the company is, where software development is concentrated). Once the transition process is defined, the transition board needs to communicate and implement the steps outlined in the process of transition.

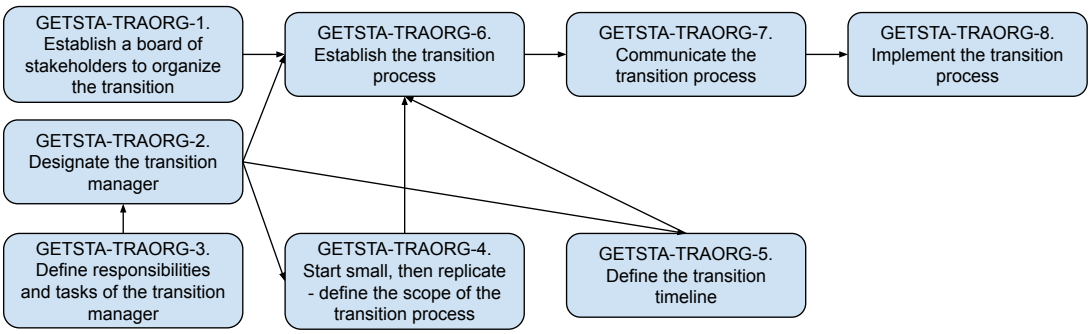


Figure 3.6: Example Process Template – Transition Organization

See Section A.1 in Appendix A for more process templates and for the complete handbook subsection with detailed best practices on transition organization.

IP-at-Risk Analysis

We identified industry best practices on analyzing potential risks of the ungoverned use of open source and ways to mitigate these risks. An overview of these practices includes:

- OSGOV-GETSTA-IPRISK-1. Run license compliance analysis
  - OSGOV-GETSTA-IPRISK-1.1. Develop standard license interpretation
  - OSGOV-GETSTA-IPRISK-1.2. Use standard license interpretation
  - OSGOV-GETSTA-IPRISK-1.3. Create license-use case pairs
- OSGOV-GETSTA-IPRISK-2. Analyze risk exposure of using an open source component
- OSGOV-GETSTA-IPRISK-3. Mitigate risk to intellectual property
  - OSGOV-GETSTA-IPRISK-3.1. Replace problematic components
  - OSGOV-GETSTA-IPRISK-3.2. Decouple problematic components
  - OSGOV-GETSTA-IPRISK-3.3. Require bill of materials for supplied code by 3rd party post-factum
  - OSGOV-GETSTA-IPRISK-3.4. Run random audits to identify previously undetected or missed open source components and their metadata
- OSGOV-GETSTA-IPRISK-4. Analyze the security risk of using an open source component

Potential risks of the ungoverned FLOSS use include open source license non-compliance, security risks and other risks to a company's intellectual property. Once the risks are identified and analyzed, companies need to mitigate them by replacing or decoupling the problematic components depending on the use case and on the license of the component used. Other mitigation practices suggest requiring detailed bill of materials for the supplied code by third parties after the delivery, as well as running random audits to identify the metadata of the previously undetected or missed open source components. Table 3.7 presents one of the best practices on the topic of IP-at-risk when getting started with open source governance.

Here is an example of a best practice from this subsection:

Name	OSGOV-GETSTA-IPRISK-1.3. Create license-use case pairs
Actor	Transition manager
Context	Your company → *developed standard license interpretation* and you are → *using standard license interpretation*. Developers are also consulting company's → *established FLOSS governance policy for the transition period*, and are contacting the transition board or the transition manager for case by case review of special cases of FLOSS use.
Problem	What's the best way to document the case by case decisions on special cases of FLOSS use, reviewed by the transition board or the transition manager?
Solution	In one centrally available document, create license/use case pairs to document the case by case decisions on special cases of FLOSS use. This document should include all the major licenses and company's detailed approach to their use in different business contexts or use cases. For example, it can be acceptable to use a copyleft license for certain (non-differentiating) products, while it might be unacceptable in other cases such as for company's main products (with competitive advantage). Such license/use case pairs should be well structured and documented. In case of a new decision on a special license/use case pair by the transition board, this document must be updated by the transition manager. Developers must consult the document before contacting the transition board or the transition manager with a new review request, because they might be able to find their answer for a specific license/use case pair in the document. Having such a document improves performance and reduces unnecessary redundancy.

Table 3.7: Best Practice OSGOV-GETSTA-IPRISK-1.3. Create license-use case pairs

The industry best practices of our theory can be traced to the data from the qualitative survey we performed. Here is an example of such a trace from the interview with Company 6's open source compliance manager talking about the specifics of open source license-use case pairs:

> *"[Our] open source handbook doesn't really present rules in a concrete setup, but what it explains all the interpretations of the licenses that we have [used]. We assess licenses with lawyers, with our internal lawyers, and from these license assessments, we determine certain [company] rules for its usage, modification, and contribution. And these rules for the individual licenses are explained in that document [as license-use case pairs]."* —*CX6.1*

Here is another example of a data trace for this best practice by a legal expert from Company 9:

> *"The first thing to recognize is that one size [of license compliance] does not fit all, there are sort of what I view as a couple of different use cases, the first most important use case is anything that gets the delivered outside the company, something that gets distributed. Why is that? Because all the copyleft licenses except the AGPL v3 depend on distribution, which is a transfer of copy to trigger the obligation. If you have a total SaaS infrastructure, you probably have a lot less risk then in a standalone application, particularly with GPL v2, as it's not a distribution. You need to match use cases with open source license interpretations."* —*CX9.1*

See Section A.4 in Appendix A for the complete handbook subsection with detailed best practices on IP-at-risk analysis.

Communication and Capabilities

We identified some meta-level best practices that enable a smooth transition towards open source governance. We group these practices under the category of communication and capabilities of our theory, including:

- OSGOV-GETSTA-COMCAP-1. Establish communication channels for open source governance issues
- OSGOV-GETSTA-COMCAP-2. Assess open source governance capabilities among developers and engineering manager
- OSGOV-GETSTA-COMCAP-3. Develop FLOSS governance and compliance capabilities at the central legal department
- OSGOV-GETSTA-COMCAP-4. Design employee training
- OSGOV-GETSTA-COMCAP-5. Provide employee training

This category of best practices covers the communication channels a company should use when getting started with FLOSS governance, as well as practices on assessing and building open source governance capabilities among developers, managers and support function employees. Building such capabilities includes employee training, as well as learning from academic literature, governance experts and organizations such as Linux Foundation[3], TODO Group[4], OpenChain[5] and SPDX[6] working groups and so on. As an example, see the best practice on designing employee training when getting started with FLOSS governance, presented in Table 3.8.

Here is an example of a best practice from this subsection:

[3]Linux Foundation – https://www.linuxfoundation.org/
[4]TODO Group (talk openly, develop openly) – https://todogroup.org/
[5]OpenChain Project – https://www.openchainproject.org/
[6]SPDX (software package data exchange) – https://spdx.org/

Name	OSGOV-GETSTA-COMCAP-4. Design employee training
Actor	Transition manager
Context	Employee training is intended to increase the awareness of the commercial use of open source and of FLOSS governance in a company. Also, training gives a common understanding of strategic and technical implications, and a common mindset on FLOSS governance.
Problem	How does one build a common understanding of FLOSS governance issues and risks? How to increase the awareness of FLOSS compliance and governance among developers and others?
Solution	Employee groups who will be involved in the training program should be initially identified (e.g. developers and architects, or team managers as well as legal team, supply chain team, etc.). All the training materials including necessary forms and documentation should be designed with the target audience in mind. For example, developers and architects receive web-based training and attend company-wide talks focused specifically on open source governance and compliance. However, company executives receive only brief information about the FLOSS governance policy. Design easy to understand but useful training. Consider repeating the key information for better retainment. After designing it, → *provide employee training* to the selected target groups.

Table 3.8: Best Practice OSGOV-GETSTA-COMCAP-4. Design employee training

See Section A.5 in Appendix A for the complete handbook subsection with detailed best practices on communication and capabilities during transition towards open source governance.

3.4.2 Inbound Governance

Key Theory Topic Overview – Inbound Governance

- Component Approval (OSGOV-INBGOV-COMAPP) - 13 best practices
- Component Reuse (OSGOV-INBGOV-COMREU) - 19 best practices

Answering the research question *RQ-TB2*, we found that after a company got started with corporate open source governance and finished the initial transition to governance, the company needs to ensure long-term inbound governance that includes processes and practices for:

- Component Search
- Component Selection
- Component Approval
- Component Repository
- Component Monitoring
- Communication
- Education.

When addressing a product development requirement, developers have an option to search for open source components to fully or partially fulfill the requirement. We found that open source software can often be cheaper than purchasing a third-party component or developing own solutions in-house. Though using open source comes with its own costs and requirements, such as complying with open source licenses, fixing bugs, and updating OSS software. Our theory's take on inbound governance deals with all these issues providing industry best practices on the above-mentioned subtopics.

We found industry best practices for searching open source components, including:

- Define component search requirements
- Check component repository before search

- Develop search recommendations
- Update search recommendations
- Follow search recommendations
- Document search channels and practices
- Search multiple channels.

After searching for open source components, companies should select the ones that match their open source governance policy and product requirements best. In a similar logic to component search, the best practices in this subsection include:

- Define component selection requirements
- Define component selection criteria for open source projects
- Define component selection criteria for open source communities
- Define component selection criteria for open source code
- Develop selection recommendations
- Update selection recommendations
- Follow selection recommendations.

Companies then need to follow open source governance processes when approving the use of a selected open source component, which includes best practices, such as:

- Define the component approval process
- File a component approval request
- Review a component approval request
- Define transparent rules for open source component approval
- Communicate open source component approval rules
- Make a component approval decision

- Appeal a component approval decision.

After an open source component is approved and used in a company product, developers need to document this use in a centralized database to ensure future reuse of the same component (under the same approved license, the same version, etc.) in the company. To reuse open source components companies need to follow proposed industry best practices, such as:

- Establish component reuse policy
- Designate a role of responsibility for the component repository, in multiple places in the company
- Establish component reuse process
- Create component repository
- Update component repository
- Provide component repository a single well-defined location
- Search component repository for reusable components.

To ensure the quality and the reliability of the open source components in company's products and in the component repository (for potential reuse), companies need to follow industry best practices for component monitoring, such as:

- Monitor and maintain components
- Monitor components for updates
- Monitor components for license changes
- Monitor component for vulnerabilities
- Monitor project for community health.

As to some of the meta-tasks of inbound governance, companies need to ensure company-wide communication around FLOSS governance processes and topics. This builds on top of the established communication during the getting started phase. Communication best practices include:

- Establish communication channels for open source governance
- Regularly use communication channels for open source governance
- Communicate goals of open source use
- Communicate risks of open source use
- Communicate governance processes
- Communicate component requirements
- Communicate trustworthy component sources.

Finally, companies aiming at establishing continuous inbound governance embedding in software development need to educate their employees (developers, managers, lawyers, etc.) on the topics and processes of inbound governance used. The following industry best practices can be used to address the issue of governance education:

- Educate new developers
- Provide employee training on strategic governance topics
- Provide employee training on specialized governance topics
- Repeat education at regular intervals.

In our study of inbound open source governance, we chose two topics of high industry relevance and academic interest to address in full detail, presenting example best practices and process template in the following subsections of the dissertation. We focused on Component Approval subtopic, which we present in Section 3.4.2. We also focused on Component Reuse subtopic, which we present in Section 3.4.2.

Component Approval

Within the scope of the inbound open source governance, one of the key aspects of our theory covered the approval of the open source components selected for the use in products. We identified a set of industry best practices for component approval. Collectively they covered the key aspects of governing component approval, including the process of approval, the rules for approval, the guidelines for decision making and escalation. The identified industry best practices on component approval include:

- OSGOV-INBGOV-COMAPP-1. Define the component approval process
- OSGOV-INBGOV-COMAPP-2. File a component approval request
- OSGOV-INBGOV-COMAPP-3. Review a component approval request
- OSGOV-INBGOV-COMAPP-4. Define transparent rules for open source component approval
- OSGOV-INBGOV-COMAPP-5. Communicate open source component approval rules
- OSGOV-INBGOV-COMAPP-6. Make a component approval decision
- OSGOV-INBGOV-COMAPP-7. Appeal a component approval decision
- OSGOV-INBGOV-COMAPP-8. Communicate component approval process
- OSGOV-INBGOV-COMAPP-9. Implement component approval process
- OSGOV-INBGOV-COMAPP-10. Provide approval request templates
- OSGOV-INBGOV-COMAPP-11. Analyze code for license compliance
- OSGOV-INBGOV-COMAPP-12. Review use in the context of product architecture
- OSGOV-INBGOV-COMAPP-13. Add decision to component repository.

Component approval is an essential part of inbound governance. Companies can prevent risks related to FLOSS use in products, if they carefully review the components selected for use in products. To ensure the consistent and transparent component approval, companies establish a well-defined

process and clear rules that employees need to follow to make component approval or rejection decisions. One of the best practices from our theory addressed the establishment of an overall process for component approval, presented in Table 3.9.

Another identified best practice covered the component approval decision making in accordance with the predefined rules of component approval companies need to establish first. Table 3.10 presents the practice on approval decision making. It also points to further practices on appealing approval decisions, as well as on reviewing the use of a component in the context of product architecture during component approval.

Here is an example of a best practice from this subsection:

Name	OSGOV-INBGOV-COMAPP-1. Define the component approval process
Actor	OSPO (Open Source Program Office)
Context	One of the key aspects of open source governance is components approval. Software developers routinely go through a process of searching, selecting, approving and integrating software components into the company's products. The same processes apply for open source components.
Problem	Using open source components has its unique complexities, such as considering open source licenses, their obligations, and interdependencies. What should an open source component approval process include to address all these considerations?
Solution	Companies using open source components in their products need to establish components approval processes that follow → *component search* and → *component selection*. OSPO must define a streamlined component approval process that does not hinder the production, but reliably ensures that the selected open source component can be used in the product without any negative side effects.

The component approval process consists of:

- *filing a request*
- *reviewing a request*
- *making a decision*
- *appealing a decision.*

The component approval process can be assisted by tools as part of the production toolchain, in order to automate the request and decision submission, and communication.

Table 3.9: Best Practice OSGOV-INBGOV-COMAPP-1. Define the component approval process

Here is another example of a best practice from this subsection:

Name	OSGOV-INBGOV-COMAPP-6. Make a component approval decision
Actor	OSPO (Open Source Program Office)
Context	Software developers → *file component approval requests* to OSPO. OSPO → *reviews component approval requests*. Now OSPO needs to make a decision whether to approve or reject the use of the given open source component in the product.
Problem	How should OSPO make a decision about component approval requests?
Solution	OSPO must first double check if the component can be automatically approved or rejected. This applies only to the previously used license/use case pairs, meaning the requested open source license has already been used in the requested use case. OSPO refers to its → *defined rules for open source component approval* and its previous → *decisions added to component repository*.

The following decisions are taken:

- if open source licenses contradict the company's open source governance policy for all use cases, then the component is automatically rejected
- if open source licenses/use case pairs contradict company's open source governance policy, then the component is automatically rejected
- if open source licenses/use case pairs correspond to the company's open source governance policy, then the component is automatically approved.

For situations where the open source license and/or the use case are new to the company, OSPO needs to → *analyze code for license compliance*, while assessing its use case. After this OSPO (supported by the legal department) must decide if the new license/use case pair corresponds to the company's open source governance policy. To decide OSPO hears the assessment of its legal and business decision maker members. OSPO also → *reviews open source component use in the context of product architecture*. Once an approval or rejection decision has been made, OSPO → *adds this decision to component repository*.
The developer who submitted the component approval request can → *appeal a component approval decision* to the Open Source Program Officer.

Table 3.10: Best Practice OSGOV-INBGOV-COMAPP-6. Make a component approval decision

The industry best practices in our theory are interconnected forming workflows or process templates that practitioners can use to apply our handbook for corporate open source governance. We present an example process template from this subsection in Figure 3.7. According to this workflow from our theory, companies need to start by defining a component approval process and transparent rules for open source component approval. The defined rules will then be used across the company in the component approval process to review component usage requests by developers to

the open source program office (or equivalent body dealing with inbound governance within the company). After reviewing a request, the OSPO then decides whether the open source component is approved or rejected for use in the given company product. If the request is rejected, developers search for an alternative way of addressing the product requirement (e.g. searching for another OSS component, purchasing a third-party component, or developing in-house). Developers also need to be able to appeal a component approval decision that would ensure a second (more detailed) round of review. If OSPO decides to approve the use of the requested OSS component, the developer can proceed and use it, while also adding the decision and the component (with its metadata) into the company's open source component repository.

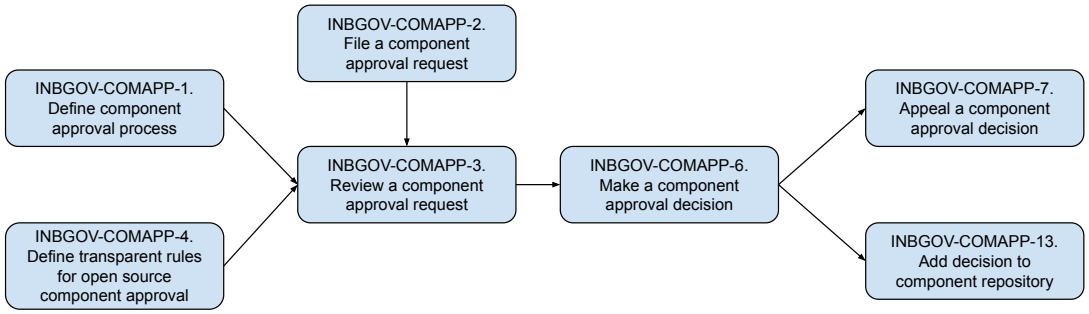


Figure 3.7: Example Process Template 1 – Component Approval

Figure 3.8 illustrates another process template from the handbook subsection on component approval. This workflow proposes another process template that companies following our governance handbook can use.

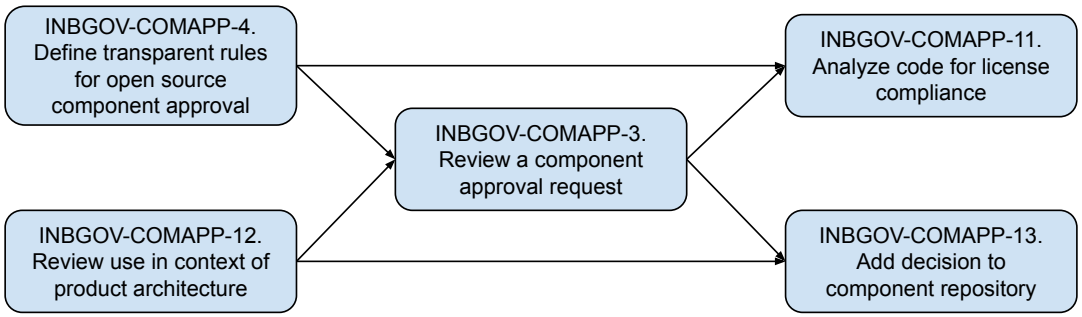


Figure 3.8: Example Process Template 2 – Component Approval

Component Reuse

After covering the component approval subtopic of inbound governance, we present our theory's take on component reuse and related governance issues, including:

- definition of a component reuse policy
- operationalization of the component reuse policy in a component reuse process
- creation and maintenance of a searchable component repository with a single well-defined location
- guidelines for auditing and using the component repository
- integration of component reuse processes with other aspects of open source governance.

For companies that are just getting started with open source governance and compliance the main problem is that developers use open source components in an uncontrolled manner, without corporate oversight or rules. This is the first problem companies need to address when getting started with open source governance, which we addressed in Subsection 3.4.1. In this subsection, we go beyond that early stage and focus on component reuse. At this stage, we assume that developers are aware of their company's approach to open source governance and are using open source components in compliance with the company's policy. The main problem at this stage is narrower; it concerns the lack of component reuse specifically. Though developers are using open source components individually, they do not share any information about their use centrally, which hinders any open source component reuse. Our best practices address this issue in detail:

- OSGOV-INBGOV-COMREU-1. Establish component reuse policy
- OSGOV-INBGOV-COMREU-2. Communicate component reuse policy
- OSGOV-INBGOV-COMREU-3. Adjust and improve component reuse policy
- OSGOV-INBGOV-COMREU-4. Designate a role of responsibility for the component repository, in multiple places in the company

- OSGOV-INBGOV-COMREU-5. Establish component reuse process
- OSGOV-INBGOV-COMREU-6. Communicate component reuse process
- OSGOV-INBGOV-COMREU-7. Implement component reuse process
- OSGOV-INBGOV-COMREU-8. Create component repository
- OSGOV-INBGOV-COMREU-9. Update component repository
- OSGOV-INBGOV-COMREU-10. Maintain component repository
- OSGOV-INBGOV-COMREU-11. Audit component repository
- OSGOV-INBGOV-COMREU-12. Use tools to create, update and maintain component repository
- OSGOV-INBGOV-COMREU-13. Provide component repository a single well-defined location
- OSGOV-INBGOV-COMREU-14. Track prior approval data for reuse
- OSGOV-INBGOV-COMREU-15. Provide all relevant metadata for component
- OSGOV-INBGOV-COMREU-16. Search component repository for reusable components
- OSGOV-INBGOV-COMREU-17. Contact OSPO for details on a repository entry
- OSGOV-INBGOV-COMREU-18. Add security check information to component repository
- OSGOV-INBGOV-COMREU-19. Link BOM and component repository

Table 3.11 presents one of the best practices on the topic of component reuse within inbound open source governance.

Here is an example of a best practice from this subsection:

Name	OSGOV-INBGOV-COMREU-5. Establish component reuse process
Actor	OSPO (Open Source Program Office)

Context After → *establishing a component reuse policy* in accordance with the company's → *defined goals of governance*, you must now define how exactly should the employees reuse open source components in the company's products.

Problem You → *established a component reuse policy* and → *communicated a component reuse policy*. However, the reuse policy is too broad to apply operationally. It focuses on the company's principles for reusing open source components, but leaves out the operational aspects of component reuse. How should you operationalize the component reuse policy, while making sure it is widely accepted and used?

Solution Specify the company's approach to reusing open source components that have already been approved by the Open Source Program Office. As open source component reuse is virtually inevitable and necessary for efficient development, a process will ensure that the current local solutions or workarounds for component reuse are replaced by a centrally defined and unified process.

Such a process has a number of benefits including, but not limited to:

- point of reference for new developers or managers who need to reuse an open source component
- a centralized database or component repository with all the used open source component across the organization
- consistent and up-to-date metadata for the used open source components
- an easy search of the open source components in use and metadata of this use, including component approval decisions, information on where these components have been used, and information on previous component owners.

The component reuse process should follow these principles:

- be clearly defined
- be easy to follow without constant guidance by the OSPO (OSPO should handle the exceptions on a case by case basis)
- be scalable and replicable
- be assisted with tools that automate and/or ensure compliance with the process.

First, you need to establish preparatory steps for component reuse process. For this, you need to:

⇒ *Establish a component reuse policy*

⇒ *Communicate the component reuse policy*

⇒ *Adjust and improve the component reuse policy*

⇒ *Designate a role of responsibility for the component repository, in multiple places in the company.*

Second, you need to establish a process for the main artefact of component reuse - the component repository, which is the company-internal database with all the used open source components, their component approval decisions and details, their metadata, and reuse information. For this, you need to:

⇒ *Create a component repository*

⇒ *Update the component repository*

⇒ *Maintain the component repository*

⇒ *Audit the component repository.*

Third, you need to establish the essential process for the component reuse. For this, you need to:

⇒ *Track prior approval data for reuse*

⇒ *Provide all relevant metadata for a component*

⇒ *Search the component repository for reusable components*

⇒ *Contact the OSPO for details on a repository entry.*

Depending on your specific needs you should modify the essential process to include more steps that would guide the employees in reusing open source components.

Table 3.11: Best Practice OSGOV-INBGOV-COMREU-5. Establish component reuse process

The industry best practices in our theory are interconnected forming workflows or process templates that practitioners can use to apply our handbook for corporate open source governance. We present an example process template from this subsection in Figure 3.9. This process template gives a comprehensive overview of component reuse at companies, starting with reuse policy, the designation of the responsible employee roles, and the establishment of a component repository for the previously used OSS components and their metadata relevant for future reuse. The component repository needs to be maintained and audited by the OSPO to ensure that the included components are up-to-date and have the correct metadata, such as open source license for the given version of the used component, relevant copyright, and export restriction information, etc. Finally, the workflow results in the operationalization of the component reuse policy in the form of the component reuse process that is communicated to the stakeholder employees, and implemented across the company.

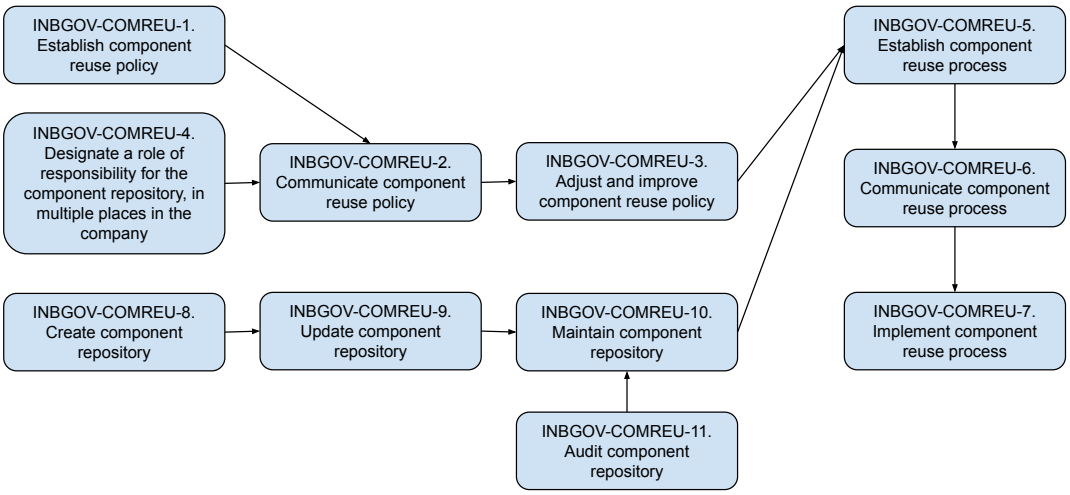


Figure 3.9: Example Process Template 1 – Component Reuse

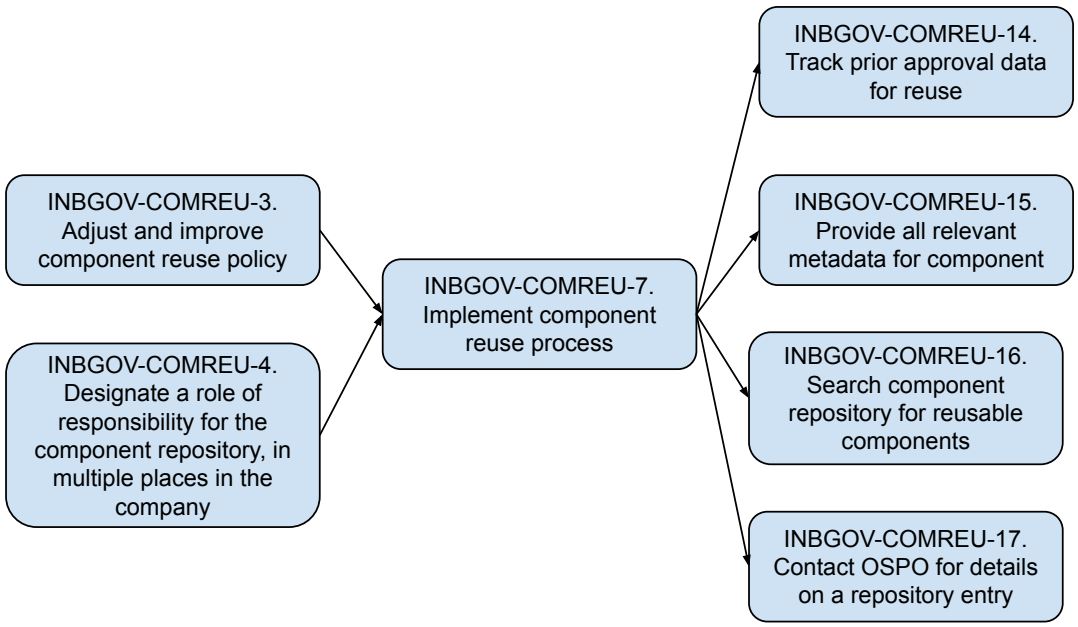


Figure 3.10: Example Process Template 2 – Component Reuse

We present another example of a component reuse process template in Figure 3.10. This workflow focuses on the specifics of component reuse with the support of the component repository. When following the component reuse process, employees should, among other tasks, track the

prior approval data, provide all relevant metadata for components, search component repository for reusable component, and contact the OSPO for details on a specific repository entry. For example, if the component had been approved for a license / use-case pair that differs from the intended reuse use-case, the developer needs to resubmit a component approval request instead of simply reusing the component approved for another use-case and added into the component repository.

See our previous work for more patterns on industry best practices for component reuse as part of inbound governance [71].

3.4.3 Supply Chain Management

Key Theory Topic Overview – Supply Chain Management

- Supply Chain Management Policy (OSGOV-SUCHMA-SCMPOL) - 3 best practices
- Supply Chain Management Process (OSGOV-SUCHMA-SCMPRO) - 5 best practices
- Preventive Governance (OSGOV-SUCHMA-PREGOV) - 4 best practices
- Corrective Governance (OSGOV-SUCHMA-CORGOV) - 4 best practices
- Bill of Materials Management (OSGOV-SUCHMA-BOMMAN) - 4 best practices
- License Compliance for Supply Chain (OSGOV-SUCHMA-LICCOM) - 2 best practices

Answering the research question *RQ-TB3*, we found that an essential part of open source governance is focused on supply chain management. Most open source components end up in company products through software supply chains. For example, if a third-party software component is purchased to be used in a company product, it is rarely checked for open source license compliance. Instead, companies often rely on supplier contracts for any potential intellectual property issues or license non-compliance of the sold company products caused by the supplied code. However, open source governance on the topics of SCM goes beyond supplier contracts and compliance checks. Being the focal topic of our theory, we go into detail of the following aspects of SCM-related FLOSS governance:

- Supply Chain Management Policy
- Supply Chain Management Process
- Preventive Governance
- Corrective Governance
- Bill of Materials Management

- License Compliance for Supply Chain.

The subsection on *Supply Chain Management Policy* proposes industry best practices for defining company-wide guidelines and high-level rules for managing software supply chains in line with the company's open source governance policy.

The subsection on *Supply Chain Management Process* operationalizes the SCM policy setting up an overall process that the company employees follow in their day-to-day tasks related to SCM open source governance.

The subsection on *Preventive Governance* proposes industry best practices that help companies prevent potential governance risks resulting from open source use by suppliers and the supplied code used in company products.

The subsection on *Corrective Governance* covers industry best practices that solve issues caused by supplier code that has open source license compliance issues or other governance issues by providing actionable advice on identifying and addressing such issues within software supply chains.

The subsection on *Bill of Materials Management* focuses on a key SCM issue of BOM management – a governance instrument that companies can use to document, track, and manage the open source components supplied to the company, as well as their metadata.

The subsection on *License Compliance for Supply Chain* covers industry best practices of open source license compliance in the context of software supply chains. An essential part of the SCM process, this subsection addressing the review of the open source license obligations caused by the supplied OSS components, as well as the fulfillment of these obligations.

In this section, we discuss the subtopics and specific best practices for supply chain management in the context of corporate open source governance. Subsection 3.4.3 on Supply Chain Management Policy presents the insights on the SCM policy. Subsection 3.4.3 on Supply Chain Management Process presents the insights on the SCM process. Subsection 3.4.3 on Preventive Governance covers the best practices for preventing governance issues caused by the software supply chains. Subsection 3.4.3 on Corrective Governance presents the insights on the corrective measures for the SCM-caused

issues that were not prevented. Subsection 3.4.3 on Bill of Materials Management presents the best practices on dealing with BOMs provided by the suppliers. Subsection 3.4.3 on License Compliance for Supply Chain covers the best practices for reviewing and ensuring open source license compliance in the context of software supply chains.

The industry best practices in our theory are interconnected forming workflows or process templates that practitioners can use to apply our handbook for corporate open source governance. We present an example process template for supply chain management in Figure 3.11. This process template presents a workflow focused on SCM policy and process. First, companies should establish the SCM policy based on their overall governance strategy, which is then followed by designating responsible employees in different parts of the company who would operationalize the policy by following the SCM process. The stakeholder employees should use tools to automate the management of software supply chains, and of the supplied code. Tools should be used for both the preventive and the corrective governance in the SCM context.

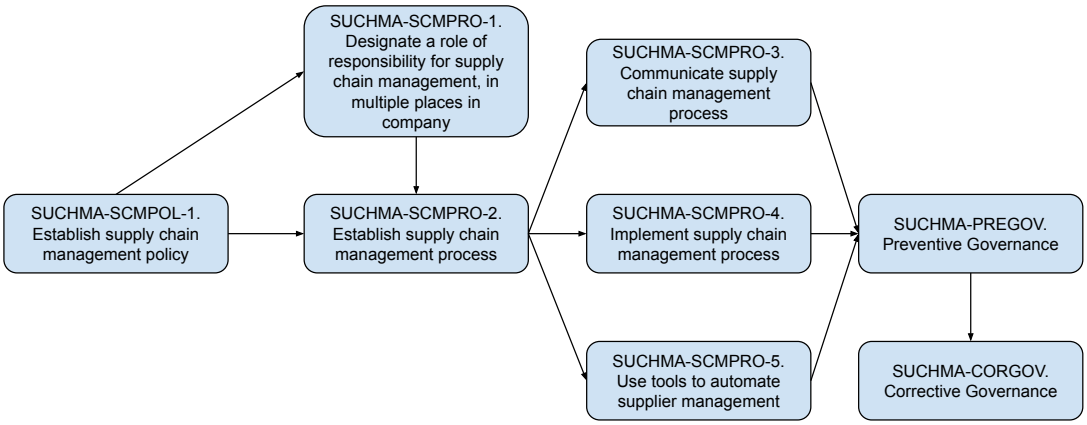


Figure 3.11: Example Process Template 1 – Supply Chain Management

We present another example process template for supply chain management in Figure 3.12. This workflow focuses on BOM management and license compliance. Companies start by identifying OSS components from the supplied software together with its metadata (e.g. open source license

data, component version, copyright information, etc.). Next, companies need to track, document, and update BOMs in a consistent and complete manner. Companies then should follow industry best practices to create a backup of open source components (hosted by the company), use a machine-readable format for BOMs, the best practices for license compliance in the SCM context.

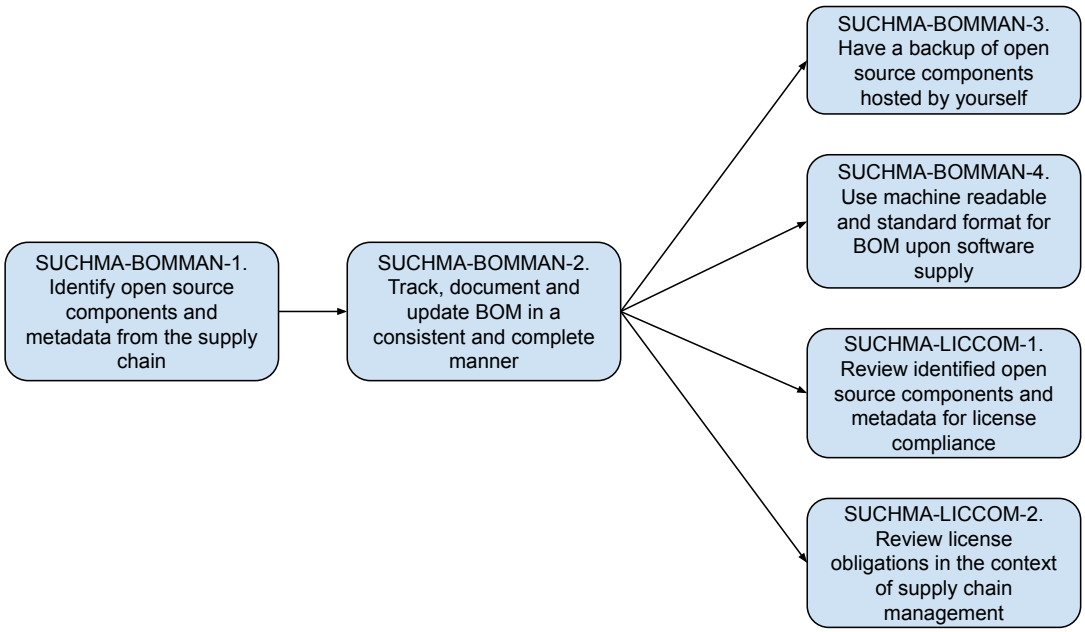


Figure 3.12: Example Process Template 2 – Supply Chain Management

Supply Chain Management Policy

We propose industry best practices for establishing, communicating, and implementing a company-wide policy or set of guidelines for managing software supply chains in terms of open source governance. The policy helps a company define its principles in regard to open source use and governance in the context of software supply chains. The policy addresses the SCM governance aspects, such as company goals for supplier management, metrics for efficient supplier management, principles for managing open source components in the supplied code, recommendations for automating supplier management through tools, rules for suppliers that use open source components, guidelines for managing supplied open source components, etc.

Best practices in this category include:

- OSGOV-SUCHMA-SCMPOL-1. Establish supply chain management policy
- OSGOV-SUCHMA-SCMPOL-2. Communicate supply chain management policy
- OSGOV-SUCHMA-SCMPOL-3. Adjust and improve supply chain management policy

Here is an example of a best practice from this subsection:

Name	OSGOV-SUCHMA-SCMPOL-1. Establish supply chain management policy
Actor	OSPO (Open Source Program Office)
Context	After → *defining goals of governance* and → *establishing an open source program*, you defined roles, responsibilities, and policies to address various aspects of open source governance in an abstract manner. Now you need to define the specific policy for managing software suppliers and open source software that you get through them either directly or indirectly. Having such a policy will help mitigate license non-compliance risks associated with the use of open source components supplied to you.

Problem Without such a supply chain management (SCM) policy you do not have a systematic way of addressing open source governance and compliance within your software supply chains. Unlike the open source components introduced by a company's own developers, supplied code can have unidentified open source components that could pose compliance risks. One of the reasons is that most software suppliers supply only a binary and not the source code. While the source code can be easily scanned for its open source components and their metadata such as licenses, binaries are harder to scan reliably and require specialized tools. Identifying open source components that have been used by the suppliers of your supplier on the next tiers of the supply chain are even harder and imprecise. Different suppliers have different levels of understanding and awareness in regards to open source licenses, compliance, and governance. This results in different compliance risk levels associated with different suppliers. The critical dependence on certain suppliers increases the complexity of such an ungoverned use of open source components supplied as part of purchased software components for used in your final products. How can you ensure open source governance and compliance while managing the complexities of software supply changes?

Solution In parallel to → *establishing the supply chain management process*, establish supply chain management policy that the company's approach to managing open source components acquired through software suppliers and not through the internal → *component approval process* supported by the Open Source Program Office. The supplier management policy presents a set of principles that guide a company through the FLOSS governance in its supply chain to help identify all the open source components used in the supplied code on all tiers of the supply chain and their metadata.

The policy often has two logical parts: the intention of the policy explaining the Open Source Program Office's motivation for having this policy, and guidelines and best practices that are later operationalized through the → *established the supply chain management process*.

In particular, the aspects covered by the policy include but are not limited to:

- company goals for supplier management
- metrics for efficient supplier management
- principles for managing open source components in the supplied code
- recommendations for automating supplier management through tools
- rules for suppliers that use open source components
- guidelines for managing supplied open source components
- supplier management integration with other aspects of open source governance, including component approval, component tracking, and license compliance
- recommended best practices for managing supplied open source components.

Select industry best practices for supplier management are presented in this handbook, such as:

⇒ *Assess open source governance and compliance awareness and maturity*

⇒ *Request supplier certification or self-certification*

⇒ *Design supplier contracts with open source governance aspects in mind*

⇒ *Audit your supply chain*

⇒ *Don't run your supplier out of business*

⇒ *Get the source code (before changing the supplier)*

⇒ *Identify Open Source Components and Metadata from the Supply Chain*

⇒ *Have a Backup of Open Source Components Hosted by Yourself*

⇒ *Use Machine Readable Bill of Materials (BOM) upon Software Supply*

⇒ *Use Standard Format for BOM upon Software Supply*

⇒ *Review License Obligations in the Context of SCM*

⇒ *Use tools to automate supplier management.*

The supplier management policy helps the Open Source Program Office define a consistent approach to the issue that can be systematically documented, implemented and communicated across the organization. It should evolve and can be modified by the Open Source Program. It should be easy to read and to the point with appropriate references to other parts of this handbook. The policy should be created in collaboration with the engineers. If the lawyers create it, it will be comprehensive and well written, but few people will ever read or follow it, because of its complexity and language. The policy should be related directly to the day to day tasks of software development and solve problems that engineers and managers face in their work.
The policy should address the principles and company approach to the key topics of supplier management:

- Supply chain management process
- Preventive supply chain management governance

- Corrective supply chain management governance
- Bill of materials management
- License compliance for supply chain.

The specific guidelines and best practices for these topics are presented in the other parts of this section. The policy is operationalized through the → *supply chain management process* that defines the steps that developers and other roles need to follow in order to efficiently manage a company's supply chain and the related open source component in their products or projects. Once the policy is established, it is necessary to → *communicate supply chain management policy* and to → *adjust and improve supply chain management policy*.

Table 3.12: Best Practice OSGOV-SUCHMA-SCMPOL-1. Establish supply chain management policy

The industry best practices of our theory can be traced to the data from the qualitative survey we performed. Here is an example of such a trace from Company 9's lawyer responsible for open source compliance talking about the specifics of establishing a SCM policy:

> *"So what you need to think about is, returning to supply chain, I think you need to manage it [in terms of open source governance], that means you need to have a policy that needs to cover [guidelines for] when you get software from third parties – a company [that] is providing you code for your product, or hardware, which has code on it. Take a look at iPhone or Apple or Android phone, the amount of code that's in there from the actual company that solves that problem [is] very small. ... many products have code from 3, 4, 5, 15 suppliers in it, right? So [as a company, you] have to make sure you have intake procedure. You [should] talk about the decision [making rules] with your programmers as well."* —*CX9.1*

Here is another example of a data trace from the primary material with an open source policy template draft by OpenChain:

> *"[We recommend that] a written open source policy exists that governs open source license compliance of the Supplied Software distribution. The policy must be internally communicated."* —*PM11.2*

See Section B.1 in Appendix B for the complete handbook subsection with detailed best practices on SCM policy.

Supply Chain Management Process

We propose industry best practices that operationalize the supply chain management policy by suggesting a step-by-step process embedded in the company's software development and software procurement processes. The SCM process guides employees, such as (software) project managers, (software) product managers, technical product managers, senior developers, and IT manager for internal tasks related to SCM. It also guides procurement managers and IT managers for external tasks related to SCM (e.g. sending supplier requests, answering supplier requests, etc.). The SCM process covers different best practices in the handbook section on SCM, such as assessing open source governance and compliance awareness and maturity of a supplier, requesting supplier certification, auditing the supply chain, etc.

- OSGOV-SUCHMA-SCMPRO-1. Designate a role of responsibility for supply chain management, in multiple places in the company
- OSGOV-SUCHMA-SCMPRO-2. Establish a supply chain management process
- OSGOV-SUCHMA-SCMPRO-3. Communicate supply chain management process
- OSGOV-SUCHMA-SCMPRO-4. Implement supply chain management process
- OSGOV-SUCHMA-SCMPRO-5. Use tools to automate supplier management

Here is an example of a best practice from this subsection:

Name	OSGOV-SUCHMA-SCMPRO-4. Implement supply chain management process
Actor	OSPO (Open Source Program Office), Supply chain management responsibles
Context	After → *establishing a supply chain management process* and → *communicating it*, you must implement the process across the company.
Problem	Implementing a large-scale process across the company has its challenges. How should you implement an efficient process?

Solution Gradually implement the supplier management process. Start by introducing preventive governance measures. Preventive governance ensures that potential suppliers have a high degree of open source governance and compliance awareness, and thus are unlikely to supply non-compliant code. Preventive governance includes the following best practices:

⇒ *Choose the right supplier*

⇒ *Assess open source governance and compliance awareness and maturity*

⇒ *Request supplier certification or self-certification*

⇒ *Design supplier contracts with open source governance aspects in mind.*

Corrective governance ensures that after the code supply any governance and compliance issues are identified are corrected and addresses, mitigating the risks caused by suppliers. Even after preventive governance, there is a potential risk in terms of FLOSS governance and compliance, so the following best practices should be applied:

⇒ *Audit your supply chain*

- *Enable regular audits*
- *Enable surprise audits*

⇒ *Mitigate identified risks*

- *Assess risks in accordance with the supply chain management policy*
- *Trigger supplier contract clauses and get the supplier to take care of the issue*
- *Don't run your supplier out of business.*

In parallel to preventive and corrective governance measures, focus on bill of materials management and on license compliance for the supply chain.

For bill of materials management, follow the best practices of the handbook to:

⇒ *Identify open source components and metadata from the supply chain*

⇒ *Track, document, and update BOM in a consistent and complete manner*

⇒ *Have a backup of open source components hosted by yourself*

⇒ *Use machine readable and standard format for BOM upon software supply.*

For license compliance for supply chain, follow the best practices of the handbook to:

⇒ *Review identified open source components and metadata*

⇒ *Review license obligations in the context of SCM*

⇒ *Review copyright notices in the context of SCM*

⇒ *Review security vulnerabilities in the context of SCM*

Depending on your specific needs you should modify the essential process to include more steps that would guide the employees in reusing open source components. Finally, the OSPO should handle the exceptions that deviate from the implemented process on a case by case basis, while considering process optimization and continuous improvement.

Table 3.13: Best Practice OSGOV-SUCHMA-SCMPRO-4. Implement supply chain management process

See Section B.2 in Appendix B for the complete handbook subsection with detailed best practices on SCM process.

Preventive Governance

We propose industry best practices for preventing potential governance and compliance issues caused by open source use in software supply chains. Companies can take a number of steps in preventing FLOSS governance issues in the context of SCM. The first preventive measure takes place when choosing suppliers. We propose a best practice for choose the right supplier taking into account the supplier's open source governance and compliance awareness and maturity. To assess the latter companies should design supplier contracts with open source governance aspects in mind, as well as consider requesting supplier certification or self-certification. Such certifications can be conducted by the company or using existing standard certification frameworks for FLOSS governance in supply chains. One example is the framework developed by the OpenChain Project[7].

- OSGOV-SUCHMA-PREGOV-1. Choose the right supplier
- OSGOV-SUCHMA-PREGOV-1.1. Assess open source governance and compliance awareness and maturity
- OSGOV-SUCHMA-PREGOV-1.2. Request supplier certification or self-certification
- OSGOV-SUCHMA-PREGOV-2. Design supplier contracts with open source governance aspects in mind

Here is an example of a best practice from this subsection:

Name	OSGOV-SUCHMA-PREGOV-2. Design supplier contracts with open source governance aspects in mind
Actor	Supply chain management responsibles, Lawyer / legal counsel, Procurement department
Context	You are → *assessing open source governance and compliance awareness and maturity* of the potential suppliers and → *request supplier certification or self-certification*.

[7]OpenChain Project – https://www.openchainproject.org/

Problem How can you use supplier contracts to address open source governance and compliance?

Solution Companies build their supplier relationships using supplier contracts with clear terms for the functionality, quality, and availability of the supplied software. However, contracts do not address aspects of legal compliance specific to open source software components, which can cause potential governance and compliance problems, as the responsibility for the final software product that uses supplied code remains with the client and not the suppliers. Even if some contracts have clauses for putting partial responsibility for legal compliance caused by suppliers, this is not enough to ensure no risk caused by ungoverned use of open source in products.

In case of potential litigation, putting the blame solely on a supplier and potentially → *running a supplier out of business is not a good strategy*, as you will be left with no supplier, no source code in most cases and a remaining legal issue of license compliance in your product. Therefore, address this issue early on and prevent risks of open source compliance and governance by adding clauses on the issue in the supplier contract as early as possible. The contract should inform the supplier of all the obligations around open source license compliance and other aspects of FLOSS governance. Design contracts that outline a supplier's responsibility in case of license non-compliance, and outline preventive measures, such as → *performing certification or self-certification*, which can be optional or mandatory. Supplier contracts can also include stricter provisions, such as specific templates that a supplier must fill before any anticipated use of open source components in software development and send these templates to the client for approval. You can then use the open source component information in the provided template to → *run through your component approval process* before allowing or rejecting the use of the component in the software that will eventually be supplied to you.

Though more cost and labor intensive this approach makes sense for companies that have critical suppliers and a large number of products that will have the supplied component. In such cases, the potential risk of non-compliance is too high to only rely on contracts and subsequent → *supply chain audit*.

Table 3.14: Best Practice OSGOV-SUCHMA-PREGOV-2. Design supplier contracts with open source governance aspects in mind

See Section B.3 in Appendix B for the complete handbook subsection with detailed best practices on preventive SCM governance.

Corrective Governance

We propose industry best practices for addressing any identified issues of FLOSS governance or compliance caused by open source use in software supply chains. Though preventive governance best practices mitigate the potential governance issues caused by lacking SCM, companies should be ready to address and correct any cases of non-compliance or other governance issues caused by suppliers. Companies should conduct regular and surprise audits of their software supply chains to find any potential issues, similar to the IP-at-risk analysis in the smaller scale during getting started with governance. Companies then need to mitigate the identified risks by assessing them in accordance to the company's supply chain management policy, triggering supplier contract clauses, and getting the supplier to take care of the issue. Industry experts recommend not running the company suppliers out of business when conducting corrective governance, as the potential losses and issues could increase in volume and complexity in case of the bankruptcy of a small supplier.

- 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

Here is an example of a best practice from this subsection:

Name	OSGOV-SUCHMA-CORGOV-2.3. 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.

Table 3.15: Best Practice OSGOV-SUCHMA-CORGOV-2.3. Do not run your supplier out of business

See Section B.4 in Appendix B for the complete handbook subsection with detailed best practices on corrective SCM governance.

Bill of Materials Management

We propose industry best practices on the key SCM issue of BOM management. BOMs are the main instrument companies can use when managing their use of open source in the supplied code that ends up in company products. BOM for the supplied software includes the list of open source components used in the supplied product, together with the relevant metadata mapped to them. Metadata includes open source license data and version, copyright information, information on export restrictions, etc. Each company should define and communicate the metadata they expect to see in the BOMs provided by the suppliers. Companies should require and use a machine-readable format and industry standards for BOMs. This enables managing multiple BOMs at once using SCM governance tools. Such tools should be also used to identify open source components and metadata from the supply chain even if a BOM has been provided, in order to ensure its correctness. Another best practices in this subsection recommends creating a backup of open source components hosted by the company, which can be useful if the supplier goes out of business, but the company continue selling a product using open source components that came from a supplier.

- OSGOV-SUCHMA-BOMMAN-1. Identify open source components and metadata from the supply chain
- OSGOV-SUCHMA-BOMMAN-2. Track, document and update BOM in a consistent and complete manner
- OSGOV-SUCHMA-BOMMAN-3. Have a backup of open source components hosted by yourself
- OSGOV-SUCHMA-BOMMAN-4. Use machine readable and standard format for BOM upon software supply

Here is an example of a best practice from this subsection:

Name	OSGOV-SUCHMA-BOMMAN-4. Use machine readable and standard format for BOM upon software supply
Actor	OSPO (Open Source Program Office), Supply chain management responsibles
Context	You have used the bill of materials and code scanning of the supplied code to → *identify open source components and metadata from the supply chain*. You have → *tracked, documented and updated BOM in a consistent and complete manner*.
Problem	What can you improve the performance of managing your BOMs?
Solution	Software supply chains are complex and cannot be handled manually. You need to → use tools to improve the performance of BOM management. Most importantly you need to establish a machine readable and standard format for BOMs. An example of such a format is called Software Package Data Exchange (SPDX). It enables the documentation and exchange of data and metadata for open source components and BOMs made of such components.

Table 3.16: Best Practice OSGOV-SUCHMA-BOMMAN-4. Use machine readable and standard format for BOM upon software supply

See Section B.4.6 in Appendix B for the complete handbook subsection with detailed best practices on BOM management.

License Compliance for Supply Chain

We propose industry best practices for open source license compliance in the SCM context. While component approval process from inbound governance focuses on license compliance of the open source components that are added into company products directly by developers during in-house developers, the supplied code does not go through such an approval process. To address the issues that were not identified during the review of the BOMs provided with the supplied software, companies should also review the identified open source components in the supplied products, as well as their metadata for license compliance. Companies should also review the specific license obligations resulting from the used open source components by the suppliers and fulfill any such obligations.

- OSGOV-SUCHMA-LICCOM-1. Review identified open source components and metadata for license compliance
- OSGOV-SUCHMA-LICCOM-2. Review license obligations in the context of supply chain management

Here is an example of a best practice from this subsection:

Name	OSGOV-SUCHMA-LICCOM-2. Review license obligations in the context of supply chain management
Actor	OSPO (Open Source Program Office), Supply chain management responsibles
Context	You have used the bill of materials and code scanning of the supplied code to → *identify open source components and metadata from the supply chain*. You have → *reviewed identified open source components and metadata for license compliance*.
Problem	What do you need to do to comply with open source licenses of the components from the supply chain?

Solution	You need to review the obligations of the identified open source licenses in order to comply with them. Different licenses have different obligations, which all have to be documented and checked with the use cases in which the components are used. Multiple licenses in one component need to be considered, as they can result in incompatible license mixtures. Use the reviewed license obligations to → *ensure license compliance*.

Table 3.17: Best Practice OSGOV-SUCHMA-LICCOM-2. Review license obligations in the context of supply chain management

See Section B.4.11 in Appendix B for the complete handbook subsection with detailed best practices on license compliance for supply chain.

3.4.4 Outbound Governance

Answering the research question *RQ-TB4*, we found that another essential part of corporate open source governance is outbound governance. Unlike the inbound governance that focuses on how open source software gets into the company products, outbound aspects of FLOSS governance address how the company's products that incorporate open source components are released and distributed. The key subtopic of outbound governance our theory identified focused on ensuring the open source license compliance for the products the company sells. While the inbound governance and supply chain management best practices help ensure open source license compliance of the resulting products, they do not eliminate the need for a license compliance check after the software development process is over and the product is ready to be released. We found industry best practices recommending companies to take over the open source license compliance artifacts (e.g. license scan reports, BOMs, etc.) from the inbound governance and SCM processes and run further license compliance checks. In a nutshell, outbound governance includes best practices for checking and ensuring FLOSS compliance before products are shipped to the customers. Outbound governance also addresses the management of employee contributions to open source communities, as well as other governance issues in relation to external parties (customers, certification bodies, regulators, etc.).

Our theory's take on outbound governance focuses on the following subtopics:

- License compliance
- Distribution Preparation
- Product Distribution
- Release Management
- Contribution Management.

Given our theory's limited focus on outbound governance, we present one encompassing best practice on the topic, in Table 3.18.

Here is an example of a best practice from this subsection:

Name	OSGOV-OUTGOV-LICCOM-1. Ensure license compliance
Actor	OSPO (Open Source Program Office), Lawyer / legal counsel
Context	In many parts of this handbook, we cover topics of open source license compliance and its integration in different processes of open source governance, namely in getting started with open source governance, component approval, component integration and reuse, and supply chain management. Open source licenses have a certain right to using the software, but they also have obligations and limitations that must be observed. For example, a copyleft license like GPL version 3 allows companies to use the open source software under this license, but enforces an obligation, such as open sourcing any derived work, that is any software created using the GPL-licensed code. For a company, this would mean sharing openly their home-developed software, which often is its intellectual property and differentiating competitive advantage. Continuing the example, if a company uses a GPL-licensed code in a product, but fails to comply with the GPL obligation of sharing the product's source code, the company would be non-compliant with the open source license. As a result, if legally challenged the company might end up paying significant penalties in court (as has happened before) or might be forced to cease and desist the distribution of the non-compliant product. Understandably, the risk of non-compliance will have a more significant effect on a company that uses non-compliant software in multiple hardware products of its own or of its client. This software includes both home developed or supply-chain acquired components that need to be compliant. Even if a supplier supplies non-compliant code that makes it to your product, it's still the client company that is liable for any license non-compliance.

Problem How should companies ensure open source license compliance?

Solution Open source license compliance is a complex topic that deals with a matrix of numerous open source licenses (about 20 prominent ones) and various use cases such as use for software development only, use as part of a standalone or integrated software product, use as part of a SaaS product, etc. Each of the license/use case pairs needs to be interpreted by a company lawyer or by the OSPO, and followed by managers and software developers.

When → *getting started with open source governance*, companies deal with license compliance by:

⇒ *Establishing a board of stakeholders to organize the transition* and → *designating the transition manager* who are tasked with ensuring open source compliance in the transition towards open source governance. This task is then shifted to the → *open source program office (OSPO).*

When → *getting started with open source governance*, companies deal with license compliance by:

⇒ *Establishing FLOSS governance policy for the transition period* and → *establishing the transition process* that define the principles and operational aspects of open source license compliance during the transition phase.

⇒ *Establishing a process of continuous reporting and assessment* that ensures the day to day license compliance for the newly added open source components that includes assessing licenses, copyright notices, export restrictions, and software supply chains.

⇒ *Developing standard license interpretation*, → *using standard license interpretation* and → *creating license/use case pairs* which all directly ensure license compliance during the transition towards governance.

In the section on → *component approval*, companies deal with license compliance by:

⇒ *Defining the component approval process* and → *defining transparent rules for open source component approval* that include instructions on dealing with open source licenses and use cases, as well as license compliance during the component approval phase.

⇒ *Providing approval request templates* that include license information for open source components such as the license name, copyright information, use case, and whether multiple licenses are mixed.

⇒ *Analyzing code for license compliance* that directly addresses license compliance during component approval.

In the section on → *component reuse*, companies deal with license compliance by:

⇒ *Establishing component reuse policy* and → *establish component reuse process* that address principles and specific tasks of license compliance during component integration and reuse.

⇒ *Designating a role of responsibility for component repository, in multiple places in company* tasking some employees with ensuring compliance when integrating or reusing components.

⇒ *Creating component repository* and → *updating component repository* with the → *provided relevant metadata for components* that include license metadata.

In the section on → supply chain management, companies deal with license compliance by:

⇒ *Establishing supply chain management policy* and → *establishing supply chain management process* defining the principles and the operationalization of license compliance in supply chains.

⇒ *Designating a role of responsibility for supply chain management, in multiple places in company* tasking some employees to deal with supplier management responsibilities including license compliance.

⇒ *Designing supplier contracts with open source governance aspects in mind* that addresses license compliance as part of supplier contracts.

⇒ *Reviewing identified open source components and metadata*, → *reviewing license obligations in the context of SCM, and* → *reviewing copyright notices in the context of SCM* that are the best practices directly addressing license compliance in the context of supply chain management.

As these open source license compliance ensuring steps are taken, companies need to identify the measures necessary to comply with various license obligations in different use cases.

These obligations defer based on different open source licenses and must be interpreted by company lawyers, then documented and shared internally. Software developers and managers must develop compliance checklists for different license/use case pairs and follow them. Such checklists will include:

- distributing copyright notices and open source license with released products
- defining compliance artifacts

- generating compliance artifacts from product architecture model
- providing compliance artifacts
- open sourcing, distributing and hosting own software, if required.

Table 3.18: Best Practice OSGOV-OUTGOV-LICCOM-1. Ensure license compliance

Some other best practices of outbound governance include:

- establishing a compliance policy
- establishing a compliance process
- disclosing all licenses used
- defining required license artifacts
- distribution preparation
- conducting source code inspections before product release
- defining product shipment checklist
- establishing a contribution policy
- establishing a contribution process
- double checking the contributions, etc.

3.4.5 General Governance

After answering the research questions *RQ-TB1, RQ-TB2, RQ-TB3, and RQ-TB4*, we found that some industry best practices go beyond any specific question and touch on the cross-cut and more general topics of open source governance, such as establishing an overall company policy for corporate open source governance, building up an open source program office, developing open source license interpretations, etc. The general aspects of open source governance in our theory addressed the following subtopics:

- Governance Management
- Open Source Program Office
- License Interpretation
- Capabilities

Governance management focuses on the general managerial aspects of corporate open source governance, such as defining the company's goals of governance, establishing an open source program that would further detail the broadly defined goals, establishing an open source program office that would then put the program into action across the company, etc. The best practices on governance management include:

- Define goals of governance
- Establish an open source program
- Establish an open source program office
- Define the role of legal counsel
- Give legal counsel veto right
- Give arbitration committee decision right
- Integrate program office in product development
- Integrate program office in mergers and acquisitions.

An example of a best practice from the Governance Management subsection is presented in Table 3.19.

Here is an example best practice from this subsection:

OSGOV-GENGOV-GOVMAN-3. ESTABLISH AN OPEN SOURCE PROGRAM OFFICE

Name	Establish an open source program office (OSPO)
Actor	Top management
Context	Your company went through a transition from ungoverned use of open source software in products to FLOSS governance. You already → *got started with governance* following this handbook.
Problem	Now you need to institutionalize the newly established FLOSS governance process that you outlined by → *defining goals of governance* and by → *establishing an open source program*. Which institution will execute the open source program?
Solution	To execute the defined goals of governance and your company's open source program, you need to establish an open source program office. It is based and built upon the already → *established board of stakeholders to organize the transition*. However, the open source program office is a permanent and company-wide body that will have a larger scope than the transition board. As a first step, create the new team/unit that will be responsible for all open source governance-related questions including license compliance checks, governance policy execution, supplier management in terms of open source governance, etc. Second, establish the organizational structure of the program office. This depends on your company size, geography, and products. In general, consider the following employees as potential members of the program office: • members of the transition board

- senior developers from different production units (can be shifting)
- engineering managers from different production units (can be shifting)
- lawyer from the central organization of the company (e.g. corporate management) with knowledge and experience in open source compliance and governance
- software architect (overlooking several production units)
- business/product managers
- software procurement officer.

The OSPO should be inclusive and transparent. The board will be taking important business, technical and legal decisions, which need to be widely communicated and enforced. The board does not have to require the full-time engagement of all the members (depends on the company). However, it is important to → *define roles, responsibilities, and policies* and to follow other best practices for the operation of → *Open Source Program Office*.

Table 3.19: Best Practice OSGOV-GENGOV-GOVMAN-3. Establish an open source program office

Open Source Program Office has a subsection of its own that focuses on the largest topic of general governance, which covers how the OSPO should function, who the involved employees should be, what processes should it coordinate, etc. The OSPO-specific best practices include:

- Define roles, responsibilities, and policies
- Provide roles, responsibilities, and policies in written form
- Match policies to actual risks
- Provide a contact for internal inquiries

- Provide a channel for whistle-blowing
- Provide a contact for external inquiries
- Collaborate with legal counsel on license interpretation
- Track industry best practices and standards
- Network to learn from others
- Engage with the community.

Another subsection of general governance – License Interpretation included the following best practices:

- Use standard license interpretation
- Develop standard license interpretation
- Use standard license compatibility matrix
- Develop standard license compatibility matrix
- Automate license identification and interpretation.

The Capabilities subsection includes the following best practices:

- Assess open source governance capabilities
- Create educational resources for capabilities building.

4

Theory Evaluation

This chapter covers the evaluation of our theory of industry best practices for corporate open source governance. Going beyond theory building, we set out to evaluate parts of our theory through a multiple-case case study at three companies. We conducted a guided implementation of parts of our FLOSS governance handbook at production-level projects at these companies. Our goal was to evaluate how our theory performs in the real-life setting of companies using open source software and following our recommendations for governance. We evaluated parts of our theory on

getting started with open source governance and on the inbound governance at Company A – our longest-running case study of two and a half years at a company with no processes or practices for FLOSS governance. We evaluated our theory on supply chain management governance at Company B – a company with understanding and processes of FLOSS governance basics, but lacking the more advanced processes and practices, such as those related to SCM. We attempted to evaluate parts of the inbound, outbound, and supply chain management governance at Company C, but failed to fulfill the evaluation, details of which we also report in this chapter.

4.1 Overview

In Chapter 3 we proposed a theory of industry best practices for corporate open source governance that was based on the qualitative data analysis of the expert interviews conducted over the course of three years. To strengthen the proposed theory we evaluated it through a multiple case case study, where we tested how our theory applies to companies with little or no open source governance in place. We presented the research process and findings of our theory evaluation in this chapter.

To evaluate our proposed theory we studied its trustworthiness to ascertain the quality of our research and findings. We followed Guba [61] [97] identifying the criteria for trustworthiness of qualitative studies, as such criteria can differ for quantitative research [59]. This resulted in the following criteria:

- Credibility
- Dependability
- Confirmability
- Transferability.

Credibility. Credibility is the degree to which we can establish confidence in the truth of our findings in the context of the inquiry. To ensure credibility, we performed two rounds of peer debriefing, together with three colleagues we reviewed this study and incorporated the feedback from

our colleagues from within our research group. Furthermore, during data collection we conducted our interviews iteratively, adjusting our semi-structured interview questions based on the company context and on our experience with earlier interviews. The latest version of the interview questions is presented in Appendix C.

Dependability. Dependability is the degree of consistency of the findings and traceability from the data to the results. We ensured dependability by collecting and saving raw interview data, documenting our qualitative data analysis in different stages of the coding and by documenting our analysis in a manner that allowed tracing each requirement in our theory to its origin in our collected data. We included numerous direct references to the expert interviews in the presentation of our research findings in Chapter 3.

Confirmability. Confirmability is the degree to which the authors are neutral towards the inquiry and their potential bias effect on the findings. Qualitative data research realized by one researcher has inherent subjectivity and bias. Even though we followed the research method constructs carefully, there was bias associated with method interpretation and application to our specific context. To address this limitation, we had a second coder analyze our data and improve our original QDA coding based on input from the second coder [99]. One second coder worked parts of the QDA re-coding on the topic of getting started with corporate open source governance. Another second coder worked parts of the QDA re-coding on the topic of supply chain management within corporate open source governance.

Credibility, dependability, and confirmability dealt with the internal aspects of the trustworthiness of our study – based on our research design and process, which resulted in our theory. However, we were not able to evaluate the transferability of our theory in the same way. Transferability is the degree to which the findings of our study hold validity in other contexts. To evaluate the transferability, we had to look at the external validity of our results – how our theory can be generalized and applied at companies with no or little corporate open source governance in place. Such an evaluation strategy has been recommended by researchers studying the trustworthiness in qualitative

research projects [138] [114]. This chapter of the dissertation focused on evaluating the transferability of our theory through the multiple-case case study we had conducted and whose results we presented here.

4.2 Research Question

As the evaluation focus was on assessing our theory's external validity, we asked the following overarching research question (Research Question – Theory Evaluation):

RQ-TE: How transferable was the proposed theory of industry best practices for corporate open source governance in the context of companies with no or little governance in place?

Our theory was developed based on the expert knowledge on the topic at the companies with an advanced understanding of open source governance. In theory evaluation we aimed at assessing how generalizable such a theory would be to companies that were not FLOSS governance experts.

The overarching research question *RQ-TE* was then operationalized with specific quality criteria. We chose several criteria by looking at academic research from various disciplines, where qualitative theories were evaluated. Namely, the applicability, relevance, understandability, and usefulness of a theory could be used to critically appraise the transferability of qualitative research [133] [11]. Bitsch [13] added another evaluation criteria – the comprehension of the theory. Other evaluation criteria included the structure, completeness, and variability of qualitative theories [114] [85].

We used the following criteria to assess transferability:

- *Completeness*
- *Variability*
- *Structure*
- *Comprehension*
- *Understandability*
- *Applicability*

- *Relevance*
- *Significance*
- *Usefulness.*

When defining the research question and the evaluation criteria, we considered looking at some of the aspects in further detail. For example, assessing different specific aspects of significance (e.g. savings of potential costs related to open source governance and compliance risks) or understandability (e.g. increased level of open source governance awareness among employees). However, given the scope of the evaluation case studies, we only expected that the case study companies would start implementing and using our governance handbook in pilot projects and that the full roll-out of company-wide governance processes would take several years (as is the case for most company-wide policies or guidelines). Therefore, we did not expect to have access to significant quantitative or qualitative data that would enable us to evaluate further specifics of the proposed theory. Instead, we focused on evaluating the transferability of the proposed theory by looking at the above-mentioned quality criteria that can be assessed at the early phase of handbook implementation and use in the production setting at case study companies.

We included further details on the research question in our case study protocol in Appendix E and in the detailed interview questionnaire we used during theory evaluation presented in Section C.3 of Appendix C.

4.3 Research Method

Once we specified the research question for theory evaluation, we designed a research approach to answer this question. The clear scope of our evaluation was on the transferability of our proposed theory. We could evaluate the internal qualities (those related to internal validity) of our theory during theory building including its credibility, dependability, and confirmability. We also aimed at addressing the external quality (that related to external validity) of transferability during theory

building by presenting thorough descriptions of the research context and our underlying assumptions [146]. Namely, for each of our proposed industry best practice for open source governance we presented the context (one of the components of a best practice pattern we used to present our theory) in which we described in detail under which conditions and assumptions a given best practice would apply.

However, as we aimed at developing a practical and applicable theory, we went on to further evaluate its transferability by studying how generalizable our findings were. We searched for companies with little or no open source governance processes and practices in place, which would be willing to implement parts of our handbook (a practical representation of the proposed theory) and let us study and evaluate this implementation. To best realize this we chose case study research as our research method. As Yin [157] suggests case study research (in comparison to other strategies such as experiments, surveys, archival analysis, or history) is a fitting research strategy for situations that:

- ask research questions in the form of *how* or *why*
- do not require control over behavioural events
- focus on contemporary events.

Our theory evaluation question did ask a *how* question, that was on how transferable the proposed theory was in the context of companies with no or little governance. Our theory evaluation did not require control over behavioural events, nor was such a controlled study possible for a theory so complex and multi-layered (in terms of having an organization-wide impact and hierarchies of stakeholders), which could not realistically be confined to a controllable environment. Finally, our theory focused on the contemporary phenomenon of corporate open source governance, as the topic has been emerging only recently, which we already discussed in Chapter 2 on the state-of-the-art of academic literature on FLOSS governance.

We used the case study research method to test a theory (theory evaluation), which corresponded to one of the research purposes a case study could have, as suggested by case study methodology

scholars [157] [24] [42]. As outlined in the case study protocol presented in Appendix E, our case study was both descriptive and explanatory. It was descriptive in resulting in detailed reports of what the initial state of open source governance at the studied companies was, as well as how companies followed and implemented the proposed industry best practices from the proposed theory. It was explanatory in presenting the reasons why certain parts of the theory were more or less complete, understandable, applicable, useful, etc., which resulted from analyzing the proposed and the actual implementation patterns of corporate open source governance at the studied companies. Another characteristic of our research method was it being a multiple-case case study with a holistic design. We studied the implementation and use of the proposed theory at pilot project teams in each of the case study companies. The employees of these teams were our main source of data during theory evaluation. Finally, we need to report that the case study companies partially funded our research through contracts with our university. We presented the details of this funding in the following Section 4.3.1 on sampling.

When designing our research strategy for theory evaluation, we looked at potential industry partners in the professional network of our research group to find companies with no or little open source governance in place that would also be interested in cooperating with us on the topic by allowing a guided implementation of our handbook in some their production projects. We would like to highlight that our intention was to guide the implementation of our theory, and not to conduct the implementation on our own. We observed how employees at the selected companies were using parts of our handbook, but we did not directly influence them. This ensured a less biased theory evaluation, and was in line with our case study research method by Yin [157]. This explicit choice of research design also meant that we could not follow another research method – action research for which we would have to be directly involved in the implementation, rather than being only observers [9] [121].

As a result of our sampling, we chose three companies for the guided implementation of our handbook, each with different level of open source governance maturity to best enable the evalua-

tion of our theory. One of the companies (Company A) had no governance processes or practices in place, which would enable us to implement the industry best practices for the basics of open source governance. Another company we selected (Company B) already covered the basics of corporate open source governance, which would enable us to implement the more advanced industry best practices, such as those focused on supply chain management in terms of open source governance. Finally, we chose Company C to be more mature than Company A, but less mature than Company B in terms of open source governance, so we could attempt to implement implement some of the other best practices from our theory, such as some aspects of general governance, outbound governance, as well as some aspects of inbound governance (namely on component reuse). We described the details of our sampling in Section 4.3.1.

In our evaluation design we decided to conduct theory building and theory evaluation in parallel. While we were conducting and analyzing expert interviews to develop our theory, we started assessing the situation of open source governance at the sampled companies for theory evaluation. Once a major part of our theory was complete, we would take it to the evaluation companies for guided implementation. We planned to present the state of open source governance before our intervention (guided implementation) to use it as a benchmark of how our theory affected the governance at the studied companies. We presented the results of these initial situation assessments at Companies A, B, and C in the subsequent sections of this chapter. While the partial implementation was running, we were extending our theory to cover further aspects of open source governance. For example, we started our theory building in October 2016 with the development of the theory for getting started with corporate open source governance. At the same time (in October 2016) we started assessing the initial situation of open source governance at Company A. In 2017 we started the guided implementation of the getting started best practices at Company A, which continued in 2018 and early 2019. In parallel, we were developing our theory's take on general governance, inbound governance, outbound governance, and supply chain management, which were later implemented at Companies A, B, and C. We presented the details of the guided implementation and our theory evaluation in the

subsequent sections on each of the case studies. Each case study focused on the respective company.

For the actual theory evaluation within each case study, we conducted 55 interviews with the stakeholders responsible for the implementation of the governance handbook at the three companies we studied. In addition to interviews, we also collected feedback from employee stakeholders, reviewed the documentation and artifacts created during handbook implementation, as well as further notes and communication records. We described the details of data gathering in Section 4.3.2.

We analyzed the data from our evaluation interviews, as well as the evidence collected through direct observation, document and artifact reviews, in order to identify how the implementation of our theory helped the case study companies establish and improve their corporate open source governance in comparison to their initial situation. We described the changes, presented the created artifacts, discussed the successful and failed experiences for different aspects of FLOSS governance in companies. A key technique we employed in theory evaluation was called pattern matching [145] [157], which allowed us to compare the proposed open source governance patterns (industry best practices) from our theory with the patterns of their actual implementation at at the case study companies. As a result of our theory evaluation, we demonstrated how our theory developed based on the expert knowledge at companies with an advanced understanding of FLOSS governance can be transferred (generalized) to companies with no or limited understanding of open source governance. We also reported what the challenges to transferability could be resulting from the analysis of the pattern matching on different parts of our theory. We described the details of data analysis in Section 4.3.3, and its results in the subsequent sections.

4.3.1 Sampling

Following the defined research method, we set out to select companies for our multiple-case case study. As our goal was conducting guided implementations of different parts of the governance handbook at different case study companies, we aimed at selecting companies with different levels of open source governance maturity. While the goal was to choose companies with no or little un-

derstanding of open source governance, we did not want to have three companies with absolutely no governance awareness. If the latter were the case we would only be able to implement and evaluate the getting started part of our proposed theory – represented as only one (first) section of the governance handbook. Our goal, therefore, was to have one company with no governance in place, coupled with further companies with basic, though limited, open source governance processes and practices in place.

We recognized that it would be challenging to assess the maturity level of open source governance at companies that were not governance experts (the opposite was less problematic, as expert companies were usually involved in various governance and compliance initiatives, so their governance maturity was publicly visible). To overcome this challenge, we tapped into the professional network of our research group (as we had done in the sampling phase for theory building), looking for companies with little or no FLOSS governance understanding and willingness to collaborate with us on this research.

Company A. Company A was the first company that demonstrated an interest in early discussions with FAU (our group representing the university). It was a large German company operating internationally in four software-intensive industries (aerospace, internet of things, metering, electronic assemblies), and using open source software in its products (aerospace systems, IoT devices, etc.). It met our sampling criteria of having no formal FLOSS governance in place. However, some employees at Company A understood the importance of corporate open source governance raising the issue internally, eventually escalating it to a cross-organizational group of high-level managers. The latter agreed that the company needs further to take action to address the issue, which led to the discussions with our research group and the subsequent project. In 2016, Company A agreed to take part in our research on corporate open source governance and decided to fund one researcher position at FAU to work on the project. This funding had its benefits, such as giving us the resources necessary (travel costs, equipment, services such as transcription, etc.) to conduct this study and ensure that the case study company will likely go through the whole research process without inter-

ruptions and delays. However, we also took the precautions necessary to ensure that the company's funding to the university wouldn't affect our study negatively (such as introducing a bias or distorting the findings of the study). Namely, we wrote a detailed statement of work (as an attachment to the contract between the company and the university), which detailed the terms of cooperation, ensuring that the company could not limit the publication of the research findings and could not affect the research process. The initial agreement was for one year (Oct. 2016 – Sep. 2017), with a potential follow-up of two more years. This first phase focused on one of the five divisions of Company A that operated in the aerospace industry – Division A.1. In this first phase of the project with Company A, the statement of work between Company A and FAU included the following major work packages including:

- WP1. Analysis of the state of the art
- WP2. Assessment of the current governance situation at Company A (mainly Division A.1)
- WP3. Identification of the scope of industry expertise.

WP1 focused on the analysis of the existing literature (both academic and practitioner articles and books) on FLOSS governance in companies using open source software in its products. This work package was in line with our research process, as we planned to start by reading, gathering, and analyzing the related literature on the topic, which would later feed into the Chapter 2 on the state of the art of this dissertation. We also shared the identified literature with our colleagues at Company A, which helped them improve their understanding of open source governance. Another outcome of this work package included the core concepts and categories of open source governance in the literature, which we used in the next steps of our study.

WP2 focused on the initial situation assessment of open source governance processes at Company A. This was the first step of the theory evaluation at each of our case study companies. Before introducing our theory of corporate open source governance in the form of a handbook with industry best practices, we reviewed the current situation of governance at a given company to confirm that there was little or no governance in place, and to analyze the extent of the company's governance

awareness and understanding. Company A, being our first, largest, and longest running case study company, underwent the most comprehensive initial situation assessment, which started during the first phase in 2016-2017 and included 18 employee interviews at Division A.1 of Company A. This work package resulted in a detailed analysis of open source governance awareness (mostly informal and very limited, confined to one team) at Division A.1, which we used as our benchmark before our theory was introduced and evaluated at the company. A summary of this analysis is presented in Section 4.4.1 of this chapter. As a result of this work package, the deliverable to Company A was a comprehensive report on the initial situation of open source governance at Divisions A.1.

WP3 focused on the identification of the scope for industry best practices of corporate open source governance. This work package was also in line with our research process, as it helped us set up the foundations for our theory building, enabling us to conduct early expert interviews at companies with an advanced understanding of open source governance. As a result of these early expert interviews, we identified the key best practice categories and topics that would become the backbone of our theory presented in Chapter 3. This theory outline represented the scope of industry expertise on open source governance corresponding to the outline of our FLOSS governance handbook, which was the deliverable to Company A. The delivery of the handbook outline at Company A in September 2019 (end of the first phase of our project with Company A) started the handbook implementation, thus the theory evaluation phase at Company A.

The first phase of our project at Company A ended in a workshop for the employees in a special, cross-organizational interest group on open source governance at Company A, and some employees from Division A.1 (the main stakeholder of the first phase). We used this workshop to present our intermediate results, to prepare Company A (and especially Division A.1) for the upcoming handbook implementation and evaluation, and to collect additional data for the study. After the completion of the first phase of the open source governance research project at Company A, we proposed a statement of work for a follow-up project that would focus on the extension of our theory and on the full evaluation of the open source governance handbook. Company A agreed to continue the co-

operation with us for one and a half years from October 2017 to May 2019 (initially planned for two years, but shortened per Company A's request). This constituted the second phase of the FLOSS governance project at Company A, which included the following work packages (numbering for work packages continued from the first phase of the project at Company A):

- WP4. Continued assessment of the current governance situation at Company A
- WP5. Theory building and handbook development for getting started with corporate open source governance in companies
- WP6. Guided implementation and evaluation of the handbook section on getting started with corporate open source governance at a pilot project at Company A
- WP7. Theory building and handbook development for inbound governance
- WP8. Guided implementation and evaluation of the handbook sections on inbound governance at Company A (possibly beyond the pilot project).

WP4 was similar to WP2 presented earlier. The only difference for the larger scope that encompassed Divisions A.2, A.3, A.4, A.5 of Company A going beyond Division A.1, which was the main stakeholder in the first phase of our cooperation with Company A.

WP5 continued our work from WP3. While in WP3 we identified the core concepts of corporate open source governance (that corresponded to the handbook outline) from the first expert interviews (first to third sampling iterations for expert interviews during theory building, see Table 3.2), WP5 focused on deriving specific best practices in the identified categories. These best practices were the core of our proposed theory, and made part of our handbook for corporate open source governance. In WP5 we focused on the evaluation of one subtopic of our theory (corresponding to the respective handbook section) – getting started with open source governance, which we presented in detail in Section 3.4.1 in Chapter 3.

WP6 was our first major work package in theory evaluation, during which we guided the implementation of our handbook's section on getting started with open source governance at a pilot

project at Company A. The implementation took place at Division A.1 (aerospace) of the company and was conducted at a small team that read the handbook section at hand and started implementing it with our guidance. We observed this implementation and analyzed it, presenting the results of this analysis in Section 4.4.2 of this chapter.

WP7 was similar to WP5, but extended our theory building and handbook development to include various aspects (but not all) of the inbound open source governance in companies.

WP8 was similar to WP6, but focused on the implementation and evaluation of inbound open source governance topic of our theory (and their respective handbook sections).

To summarize the first company in our sample, Company A was a large German company operating internationally in four software-intensive industries. We interviewed Company A employees from Germany (Hessen, Bavaria, Baden-Württemberg), China, Mexico, Poland. We aimed at including regions outside of Germany to ensure a more representative sampling (though German companies were more accessible to us due to our university being located in Germany, and our network included more German companies). Company A had no formal open source governance in place. Case Study A at Company A was our largest and longest running study with a duration of 2.5 years in total. We evaluated the following parts of our proposed theory at Company A:

- Getting Started
- Inbound Governance.

Company B. Company B was the second company we chose for our theory evaluation case study. After the first sampling iteration (that led to choosing Company A), we recognized that the evaluation of more advanced governance topics (such as our focal topic – supply chain management) would be limited as Company A had no experience with corporate open source governance, which meant we would have to start with the very basics of governance there. Therefore, to choose the second company, during our sampling we aimed at finding a company that already had the basics of open source governance covered, but was lacking governance processes and practices for supply

chain management (a more advanced topic of governance in comparison to getting started, for example). To find such a company we were looking for a company that demonstrated an interest in the topic of supply chain management governance and collaboration with a university on the topic. Unlike with Company A, in this case we did not initiate the cooperation directly using our research group's professional network. Instead, in March 2017 we applied to an industry-university collaboration program called Software Campus[1] funded partially by the Germany industry and partially by the German Federal Ministry of Education and Research (BMBF)[2]. We submitted a project proposal that focused on the theory evaluation of supply chain management governance aspects from our proposed theory. Going through a lengthy selection process, our project was selected for funding and matched with an industry partner – a company that showed interest in the proposed collaboration, which entailed their participation in our theory evaluation case study. After an initial contact and discussion with the company, we confirmed that Company B met our sampling criteria for the second case study. It had basic corporate open source governance in place, but lacked governance in supply chain management, which was the focal topic we wanted to evaluate at the second case study company after having covered the more basic (getting started) aspects at Company A. The Software Campus project helped us partially fund our research, while also ensuring the commitment of the company to take part in our case study in the course of the project. The study was scheduled to run for one year in June 2018 – May 2019 (with some reporting and publishing activities going on until September 2019). During the project we defined and completed the following work packages:

- WP1. Analysis of the state of the art
- WP2. Assessment of the current governance situation at Company B (in terms of overall corporate governance, supply chain management in particular, and related tooling)
- WP3. Theory building and handbook extension for supply chain management within corporate open source governance

[1] Software Campus – https://softwarecampus.de/
[2] Federal Ministry of Education and Research (BMBF) – https://www.bmbf.de/

- WP4. Guided implementation and evaluation of the handbook section on supply chain management and related tooling within corporate open source governance at a production project at Company B.

The work packages were similar to those for the project at Company A presented earlier, but shifted the focus of our theory building (handbook extension) and evaluation to the topic of supply chain management as part of corporate open source governance. As a result, in WP1 we reviewed more literature to identify the key topics of governance for software supply chains, which helped us extend our understanding of the state of the art. In WP2 we assessed the governance processes and practices at Company B before our intervention (introduction of the industry best practices handbook). In WP3 we continued our theory building conducting more expert interviews with a focus on the governance for software supply chains, which were analyzed in parallel. As a result, we identified and documented industry best practices for supply chain management within corporate open source governance. We then added these best practices to our handbook, which was introduced to Company B in WP 4. In this last phase, we observed how the supply chain management section of our handbook was used at Company B in a production environment, which best practices were used and how. As at Company A, we guided the implementation but did not drive it allowing an unbiased implementation of our theory, which would lead to an impartial theory evaluation. Three months after the handbook implementation, we returned to the company to conduct follow-up interviews that we used in analyzing and reporting the theory evaluation at Company B.

To summarize the second company in our sample, Company B was a large German company operating internationally in enterprise software industry, and extensively using open source software in its products. We interviewed Company B employees from Germany (Hessen). Unlike Company A, Company B had a centralized team dealing with open source governance and compliance that worked with us in the scope of this project. Therefore, we did not study the individual divisions of the company directly like we did at Company A, which did not have any centralized governance structure. Company B already had a process of basic open source governance in place, which met

our sampling criteria for the second case study. It enabled us to evaluate a more advanced part of our theory. Case Study B at Company B had a duration of one year (plus several months for follow up and reporting). We evaluated the following parts of our proposed theory at Company B:

- Supply Chain Management.

Company C. Company C was the third company we chose for our theory evaluation case study. After two sampling iterations (that led to choosing Company A and B), we recognized that parts of our theory have not yet been evaluated. Namely, general governance, outbound governance, and some parts of inbound governance (focused on component reuse and repository) lacked evaluation. We, therefore, aimed at finding a third company to become our Case Study C, where we would implement and evaluate the above-mentioned parts of the proposed theory of industry best practices for corporate open source governance. After consulting the list of companies in our professional network, in late 2017 we contacted several companies that had expressed in potential collaboration on the topic. We prepared and sent statements of work with predefined work packages to two companies. One of the companies (our research group had previously collaborated with) reacted positively, so we had several introductory meetings to evaluate their state of open source governance and their fit in our sampling. We were looking for an internationally operating company that had some understanding of open source governance (putting it at a more mature level than Company A), while not having formally institutionalized company-wide governance (putting it at a less mature level than Company B). We found such a match at Company C, which had some ongoing initiatives around basic open source governance in individual divisions and teams, but was struggling to move forward towards company-wide governance (a focal topic of general governance from our theory). Going through a lengthy negotiation and setup process, we came to an agreement with Company C, which would fund half a researcher position at FAU to work on the project and become a case study in our evaluation study. The study was scheduled to run for nine months in April 2018 – December 2018 (with a planned extension of several months in early 2019). During the project we defined the following work packages:

- WP1. Analysis of the state of the art
- WP2. Assessment of the current governance situation at Company C (in terms of overall corporate governance, general governance, inbound governance, and outbound governance in particular)
- WP3. Theory building and handbook extension for general governance, inbound governance, and outbound governance
- WP4. Guided implementation and evaluation of the handbook sections on general governance, inbound governance, and outbound governance at production projects at Company C.

The work packages were similar to those for the projects at Company A and B presented earlier, but focused on other aspects of the theory building (handbook extension) and evaluation, including general governance, inbound governance (component reuse in particular), and outbound governance. As a result, in WP1 we reviewed further literature to identify the key topics of general governance, inbound governance (component reuse in particular), and outbound governance, which helped us extend our understanding of the state of the art. In WP2 we assessed the governance processes and practices at Company C before our intervention (introduction of the industry best practices handbook). In WP3 we continued our theory building conducting more expert interviews with a focus on the governance for general governance, inbound governance (component reuse in particular), and outbound governance, which were analyzed in parallel. As a result, we identified and documented more industry best practices for corporate open source governance on the above-mentioned topic. We then added these best practices to our handbook, which we presented to our partner team at Company C in WP 4. In this last phase, we planned to guide the implementation of the select handbook sections and best practices, observing and evaluating the application of our theory as part of Case Study C. However, unlike Case Study A and B, we encountered a problem at Company C, which led to a failed implementation and evaluation of our theory. One of the reasons was the mismatch in expectations of what a corporate open source governance handbook should

look like and be used at the company. Another reason was the pushback from the company internal team working on the issues of open source governance, as well as a pushback from company management. We discuss further details of this failed evaluation in Section 4.6.2 in this chapter. As a result, our project ended in December 2018, and we were not able to fully evaluate how our theory was used at Company C. We present the partial results from Case Study C in Section 4.6.

To summarize the third company in our sample, Company C was a large German company operating internationally in the automotive industry, and using open source software in its products. We interviewed Company C employees from Germany (Bavaria), France, USA. Unlike Company A, Company C had some understanding of open source governance, but lacked a company-wide centralized team dealing with open source governance and compliance in comparison to Company B. Such characteristics matched our sampling criteria in the scope of this project. Similar to Company A, we studied individual divisions of the company directly to assess their governance situation before in the early phase of our study. We aimed at evaluating some parts of our theory that were not studied as part the other case studies. Case Study C at Company C had a duration of nine months. We evaluated the following parts of our proposed theory at Company C:

- General Governance
- Inbound Governance (Component Reuse in particular)
- Outbound Governance.

4.3.2 Data Gathering

In the course of our multiple-case case study, we collected data from multiple sources at three case study companies, which included systematic interviews with employees in different roles (e.g. software developers, managers, lawyers), documentation, communication records, and implementation artifacts. Within each case study, we conducted semi-structured interviews with the stakeholders responsible for the implementation of the governance handbook, and with the employees using the

handbook. An overview of the conducted interviews is presented in Table 4.1. Beyond the systematic interviews, we used the following techniques (recommended by Yin [157]) for data gathering:

- conducting company-internal workshops to present the handbook to stakeholders employees, gathering their feedback and common questions
- observing directly the handbook implementation process (covering actions in real time)
- observing participants of the handbook implementation process
- reviewing company-internal documents created during handbook implementation
- reviewing the communication (emails, meeting minutes, notes) records we had with the stakeholder employees during the guided implementation of the handbook
- reviewing artifacts created by following the best practices from our handbook.

First, we conducted situation assessment interviews, analyzed them, and guided the implementation of parts of our handbook on corporate open source governance – our theory presented in the form of best practice patterns. Upon the handbook implementation, we conducted follow-up evaluation interviews to assess whether and how our theory did in a real-life setting in the case study companies. Table 4.1 gives an overview of the interviews we used for theory evaluation, including the roles of the employees we interviewed.

Figure 4.1 illustrates the case study companies we selected in our sampling – Company A in blue, Company B in red, and Company C in dark yellow; the countries of the interviewees color-coded based on the number of interviews per country; and the individual interviewees in different locations of the three companies with larger circles of a relative size to the number of interviewees at a given location.

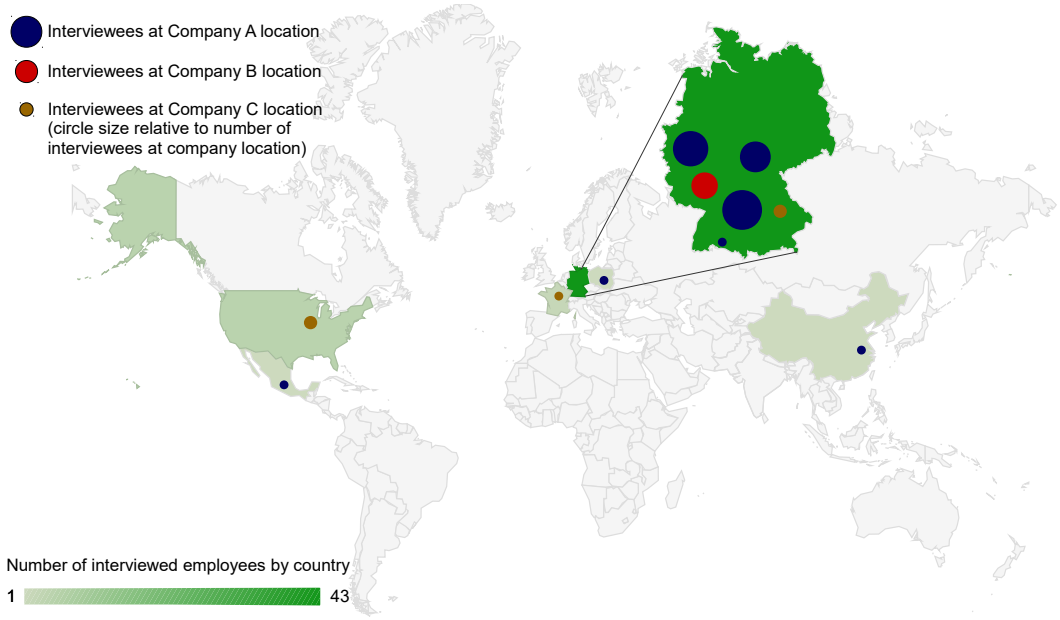


Figure 4.1: Theory Evaluation – Map of Interviewee Countries

Table 4.1 presents the overview of the main data sources we used in theory evaluation – the employee interviews at the case study companies A, B, and C.

ID	Company	Division	Employee Role	Date
CA1.1	Company A	Divisions A.1	Project Manager	2017-03-13
CA1.2	Company A	Divisions A.1	Project Manager	2017-03-13
CA1.3	Company A	Divisions A.1	Technical Manager	2017-03-14
CA1.4	Company A	Divisions A.1	Technical Manager	2017-03-14
CA1.5	Company A	Divisions A.1	Product Tester	2017-04-10
CA1.6	Company A	Divisions A.1	R&D Developer	2017-04-10
CA1.7	Company A	Divisions A.1	Tool Developer	2017-04-11
CA1.8	Company A	Divisions A.1	R&D Developer	2017-04-11
CA1.9	Company A	Divisions A.1	Product Developer	2017-03-16
CA1.10	Company A	Divisions A.1	Product Developer	2017-03-16

ID	Company	Division	Employee Role	Date
CA1.11	Company A	Divisions A.1	Product Tester	2017-03-23
CA1.12	Company A	Divisions A.1	Project Manager	2017-03-29
CA1.13	Company A	Divisions A.1	R&D Manager	2019-05-23
CA1.14	Company A	Divisions A.1	Project Manager	2019-05-23
CA1.15	Company A	Divisions A.1	R&D Developer	2019-05-23
CA2.1	Company A	Divisions A.2	Technical Manager	2017-06-28
CA2.2	Company A	Divisions A.2	Product Developer	2017-06-28
CA2.3	Company A	Divisions A.2	Division CEO	2017-06-29
CA2.4	Company A	Divisions A.2	Product Developer	2017-06-29
CA2.5	Company A	Divisions A.2	Product Developer	2017-06-29
CA2.6	Company A	Divisions A.2	Tool Developer	2017-07-06
CA2.7	Company A	Divisions A.2	Tool Developer	2017-07-06
CA3.1	Company A	Divisions A.3	Technical Top Manager	2017-09-06
CA3.2	Company A	Divisions A.3	Product Developer	2017-09-06
CA3.3	Company A	Divisions A.3	Product Architect	2017-09-06
CA3.4	Company A	Divisions A.3	Technical Top Manager	2017-09-06
CA3.5	Company A	Divisions A.3	Project Manager	2017-09-07
CA3.6	Company A	Divisions A.3	IT Manager	2017-09-07
CA3.7	Company A	Divisions A.3	IT Officer	2017-09-07
CA4.1	Company A	Divisions A.4	Product Developer	2017-10-19
CA4.2	Company A	Divisions A.4	Technical Manager	2017-10-19
CA4.3	Company A	Divisions A.4	R&D Developer	2017-10-19
CA4.4	Company A	Divisions A.4	IT Officer	2017-10-20
CA4.5	Company A	Divisions A.4	Product Manager	2017-10-20
CA4.6	Company A	Divisions A.4	Technical Top Manager	2017-10-20

ID	Company	Division	Employee Role	Date
CA4.7	Company A	Divisions A.4	Technical Top Manager	2017-10-20
CA5.1	Company A	Divisions A.5	Legal Counsel	2018-04-05
CA5.2	Company A	Divisions A.5	Division CEO	2018-04-19
CB1.1	Company B	Divisions B.1	Tool Developer	2018-10-28
CB1.2	Company B	Divisions B.1	Program Manager	2018-10-28
CB1.3	Company B	Divisions B.1	Product Manager	2018-10-28
CB1.4	Company B	Divisions B.1	Technical Top Manager	2019-02-22
CB1.5	Company B	Divisions B.1	Procurement Officer	2019-02-22
CB1.6	Company B	Divisions B.1	Compliance Manager	2019-02-22
CB1.7	Company B	Divisions B.1	Compliance Officer	2019-02-22
CB1.8	Company B	Divisions B.1	Procurement Officer	2019-05-29
CB1.9	Company B	Divisions B.1	Compliance Manager	2019-05-29
CB1.10	Company B	Divisions B.1	Compliance Officer	2019-05-29
CC1.1	Company C	Divisions C.1	OSS Release Manager	2018-09-05
CC1.2	Company C	Divisions C.1	Compliance Manager	2018-09-06
CC1.3	Company C	Divisions C.1	Product Developer	2018-09-12
CC1.4	Company C	Divisions C.1	Open Source Manager	2018-08-14
CC1.5	Company C	Divisions C.1	Legal Counsel	2018-09-14
CC1.6	Company C	Divisions C.1	IP Counsel	2018-09-18
CC1.7	Company C	Divisions C.1	Technical Manager	2018-10-24
CC2.1	Company C	Divisions C.2	Technical Manager	2018-09-25

Table 4.1: Theory Evaluation Data Sources – Case Studies

During data gathering, we followed the predefined case study protocol presented in Appendix E. For the theory evaluation interviews, we created and iteratively improved two interview question-

naires – one for the initial situation assessment of open source governance at companies, another one for the theory evaluation at case study companies after the open source governance handbook implementation. For the final versions of these two questionnaires, see Section C.2 of Appendix C.

In the initial situation assessment phase we asked the following types of questions (building upon Yin's recommendations for case study protocol questions [157]):

- Level 1: questions on specific interviewees and their context – on the interviewee roles, their involvement with open source use and governance (with different sets of questions for the employees in management teams and those in engineering teams)
- Level 2: questions about case study companies (individual cases) – on the current state of open source use at individual case study companies, main use cases and challenges of open source use
- Level 3: questions about patterns of findings across multiple case studies – on the current ways of dealing with different aspects of open source governance (e.g. current practices for inbound open source governance, for supply chain management) in the context of how other companies are addressing similar issues
- Level 4: questions about the entire case study – on the state-of-the-art practices of open source governance from the literature, the potential applicability of such practices as part of the case study
- Level 5: questions about policy recommendations and conclusions – on the views of the interviewees on potential company policies and solutions to different issues of open source governance based on their experience, current workaround, and general governance awareness.

In the theory evaluation phase after the open source governance handbook implementation we asked the following the same types of questions, but with a different focus to address the goal of assessing the implications of implementing and using our theory (and not the focus on the initial situation assessment):

- Level 1: questions on specific interviewees and their context – on the interviewee roles, their involvement with the implementation and use of the open source governance handbook (a representation of our theory of industry best practices for open source governance)
- Level 2: questions about case study companies (individual cases) – on the specifics of the handbook implementation as a whole at individual case study companies
- Level 3: questions about patterns of findings across multiple case studies – on the patterns of how companies implemented select handbook sections and best practices in the context of how other companies addressed handbook implementation
- Level 4: questions about the entire case study – on the proposed handbook and best practices for open source governance, benefits and problems with their applicability as part of the case study
- Level 5: questions about policy recommendations and conclusions – on the views of the interviewees on the recommended open source governance practices and processes, their implementation and its impact on case study companies, lacking aspects and overall handbook evaluation (not covered with the questions of other types).

Beyond the interviews, some of the other data sources included our notes and meeting minutes gathered during our communication (e.g. telcos, in-person meetings, emails) with each of the case study companies, as well as the artifacts that were developed in the course of our case studies – artifacts created and used when implementing best practice recommendations from our theory. For select examples of such artifacts for each of the case studies, see Appendix F.

We followed Yin's recommendations for data collection [157], such as using multiple sources of evidence for the concepts we were studying (this enabled data triangulation during data analysis), and creating a case study database (this enabled keeping track of multiple data sources and artifacts).

4.3.3 Data Analysis

In the course of our multiple-case case study, we started by outlining an analytical strategy, which we summarized in the case study protocol developed before the case studies began, presented in Appendix E. This strategy guided us through the data analysis at all the stages of the case study. As our case study was used for theory evaluation, we based our analytical strategy on our proposed theory of industry best practices for open source governance. This approach is in line with one of the common analytical strategies recommended by Yin [157] – relying on theoretical propositions. This strategy follows theoretical propositions that led to the case study. In our case, the theoretical propositions from our theory corresponded to the specific industry best practices from our theory for open source governance (and the proposition that applying our theory can help companies establish and improve their FLOSS governance). The objective to evaluate our theory in real-life setting at companies with no or little open source governance led to this case study (and its design). This, in turn, reflected on the research question we asked, the data we collected, and the analysis we conducted.

We analyzed the collected data iteratively. We started by analyzing each case study individually, before conducting the cross-case analysis. At each of the case studies we split the analysis into two phases (in line with data gathering presented in Section 4.3.2:

- *Phase 1*: initial situation assessment of open source governance
- *Phase 2*: theory evaluation after the open source governance handbook implementation.

In the first phase, we used the data collected at each case study company before we introduced the handbook. We analyzed the collected semi-structure interviews, notes, and other data to find out the strengths, weaknesses, opportunities, and threats of the use of open source at each company. We then analyzed the initial governance situation at each case study company. We outlined the key findings of this phase in a comparable format, using bullet point lists with the similar concepts of FLOSS governance across the case studies.

In the second phase, we used the data collected at each case study company after we introduced the handbook and guided its implementation. We analyzed the collected semi-structure interviews, notes, and other data (from participant observation, direct observation, artifacts, etc.) to derive the patterns of open source governance employed at each studied team at the case study companies. To identify and to codify these patterns touching various aspects of the proposed theory, we analyzed the implementation of the select sections of the corporate open source governance handbook, as well as on the select best practices from these sections. Discussing the handbook sections and individual best practices that were applied and used at case study companies, we assessed how they improved the state of governance at each case study company compared to the respective initial governance situations.

Following our analytical strategy, our analysis priorities were based on the theory propositions we set our to evaluate. Namely, we aimed at analyzing how generalizable our theory would be assessing different criteria of its transferability as mentioned in the research method in Section 4.3. To do so, we looked at how:

- the handbook as a whole (a practical representation of our proposed theory) can be applied at companies with no or little open source governance
- the select sections of the handbook on different core topics of open source governance (e.g. getting started, inbound governance, supply chain management) can be applied at companies with no or little open source governance
- the select best practices from different core topics of open source governance (e.g. best practice for getting started A.3.9 Run open source use analysis in products) can be applied at companies with no or little open source governance.

When analyzing the above-mentioned parts of our theory evaluation, we derived patterns of how our theory was applied at the case study companies. This was done following an analytical technique called pattern matching [145] [157]. In a nutshell, this technique matches the proposed

patterns (based on a predefined theory) with those that emerge during the case study. For our theory evaluation, the proposed patterns were based on our theory's propositions for industry best practices for corporate open source governance, which were matched with the patterns of how employees actually applied our theory at their companies. We found such patterns for the handbook as a whole at Companies A and B, patterns for the getting started and inbound governance sections of the handbook at Company A, and patterns for the supply chain management at Company B. The evaluation at Company C failed, which we discuss in Section 4.6.2. We then derived theory implementation patterns for select best practices from our theory such as:

- best practices for getting started A.3.9 (Run open source use analysis in products) and A.3.4 (Select and use governance tools for automation) at Company A
- best practices for supply chain management B.3.2 (Assess open source governance and compliance awareness and maturity) and B.3.3 (Request supplier certification or self-certification) at Company B.

To report the results of the data analysis, we presented the evaluation of the select parts of the proposed theory in respective subsections in each case study report talking about the guided implementation, created company-internal artifacts, and proposed industry best practices (from our theory) used at each company. We shared handbook implementation artifacts illustrating how our theory was applied in real-life projects. We then discussed the implementation patterns in detail. We then showed the results of our data analysis using the theoretical instrument pattern matching, presenting our observations in how case study companies were using our handbook (as a whole and in parts), identifying the patterns that emerged at companies, and comparing them with the best practice patterns we proposed initially in our theory. As a result we studied and reported the deviations between these patterns, discussing such deviations.

Furthermore, we present the results of our analysis of the predefined quality criteria for our theory evaluation. We discussed these evaluation criteria for the implemented and studied sections of the handbook (representing parts of the proposed theory) and specific best practices in each of these

sections. We first reported our analysis for the case studies individually in Chapter 4, then discussing them side by side, presenting the common findings and differences across the cases in Chapter 5. We also discussed the effects and the shortcoming of using our theory, aiming at a critical theory evaluation as a result.

4.4 Case Study A

Case Study Profile – Company A

Summary:	Large German company operating internationally in four software-intensive industries (aerospace, internet of things, metering, electronic assemblies), and using open source software in its products (aerospace systems, IoT devices, etc.)
Duration:	2.5 years (October 2016 – May 2019)
Location:	Germany (Hessen, Bavaria, Baden-Württemberg), China, Mexico, Poland
Maturity:	No formal open source governance in place
Evaluation:	Industry best practices for open source governance in Getting Started, Inbound Governance

In Case Study A, we studied the following Divisions of Company A, presented with their respective industry domains:

- Division A.1 – Aerospace
- Division A.2 – Internet of Things
- Division A.3 – Metering
- Division A.4 – Electronic Assemblies
- Division A.5 – Information Technology

From October 2016 – May 2019, we extensively studied the open source use and governance across Company A. Our first and major focus was Division A.1 that served as a pilot project in Case Study A. We conducted 12 two-hour interviews with managers, software developers and other stakeholders at Division A.1 (in different locations in the German federal states of Hessen, Baden-Württemberg, and Bavaria) using the questionnaire attached in Section C.2 in Appendix C.

Using Division A.1 as a benchmark, we went on to assess open source use and governance situation in other Divisions of Company A, namely in Division A.2 (based in Baden-Württemberg and

Bavaria), Division A.3 (based in Bavaria), Division A.4 (based in Baden-Württemberg), and in Division A.5 (based in Bavaria). For situation assessment, we interviewed:

- 12 employees from Division A.1
- 7 employees from Division A.2
- 7 employees from Division A.3
- 7 employees from Division A.4
- 2 employees from Division A.5

Note that Divisions A.5 was an internal IT-service provider with no external customers, therefore it was the smallest of all the studied divisions. It collaborated with the IT departments in the other divisions, while providing centralized support and guidance. As we had interviewed employees with IT roles in Divisions A.1, A.2, A.3, and A.4, we decided to interview only two employees at Divisions A.5 – the Legal Counsel and the Division CEO. See Table 4.1 for more details on the interviews.

At the same time we analyzed the relevant documents provided by our partners at the Divisions of Company A. We summarized our initial situation assessment for Company A as a whole in Section 4.4.1, followed by subsections on the situation assessment in individual divisions of Company A. We conducted an in-depth analysis for Division A.1, which was more extensive as that for the other Divisions. We presented the results of our analysis for Divisions A.1, followed by shorter summaries of the situation assessments for the other divisions.

We evaluated two parts of our proposed theory at Company A (mainly at Division A.1) – industry best practices for open source governance focused on getting started with governance and on inbound open source governance. We guided the implementation of our open source governance handbook at Company A, which was followed by the theory evaluation of the handbook sections on getting started and inbound governance, as well as select best practices from these sections. We report on our theory evaluation focused on getting started with corporate open source governance in Section 4.4.2 of this chapter. We report on our theory evaluation focused on inbound governance in Section 4.4.3 of this chapter.

4.4.1 Initial Situation Assessment

Confirming our sampling criteria for Company A, we found that the company and its divisions had no open source governance in place. Some informal governance existed as a way to address key issues of open source use, such as informal processes of clarifying open source license compliance when using open source components or libraries. Some employees took on the informal role of open source program office or compliance officers across the company providing support to their colleagues in their teams, divisions, and beyond. Reporting the initial situation assessment at Company A, we presented the details of the strengths, weaknesses, opportunities, and threats of using open source in Company A's products. We presented further results of our initial analysis focused on different aspects of open source governance. We then presented our initial situation at Division A.1 (our pilot project), followed by that at Divisions A.2, A.3, A.4, and A.5 in their respective subsections.

Strengths, Weaknesses, and Opportunities of Open Source Use in Products

Use of open source components in products has a number of strengths, weaknesses, opportunities, and threats (SWOT) at Company A. Before getting into the detailed threat analysis, we will highlight the three former components of the SWOT analysis, namely:

- Strengths
  - Open source software is quickly available, high quality and low cost software
  - Change in companies' software needs and culture moves industries towards more open source use in order to increase reuse of non-differentiating software, when possible
  - Major open source components and standards are universally accepted, widely tested, highly secure and well maintained by professional communities
  - Software developers in companies are interested in using more open source software in products, as they often have background and competency in open source (from university, past employers or personal projects)

  - No company resources need to be leveraged for external open source communities to use their software, except for resources to ensure compliance
  - There is a handful of commonly used open source licenses, which makes legal interpretation easier compared to different proprietary licenses for each commercial software
  - Source code for open source software is freely available, thus independent from open source community activity, companies using such software can access the source code directly, make changes and updates, unlike commercial software access to which depends on the company selling such software (especially problematic when a company goes out of business, but its software is still used in products).

- Weaknesses
  - General
    * OSS governance processes are not defined and documented
    * Rough transition from R&D to production due to OSS used
  - License Compliance
    * Legal department does not provide guidelines/checklists to developers
    * License identification and compliance check are done manually
    * License compliance decisions are not shared internally
  - Component Search
    * No defined value-effort estimation is realized, but a rule of thumb decision
    * There is no centralized internal repository to search for used OSS components
  - Component Selection
    * Technical manager decides which component to use without formalized value-effort estimation
    * There is no universal / standard template for OSS component presentation
  - Component Reuse
    * No centralized repository with OSS components and license data

  - OSS Capabilities
    * Lacking OSS competency at legal department (bottleneck for production)
    * Lacking OSS competency and support at IT department (limited development tools available)
    * OSS capabilities are concentrated around several developers
  - OSS Contribution
    * Redundant versioning (repatches to new versions of OSS components)
    * Less functional OSS use (without updates).

- Opportunities
  - Using more OSS can further decrease software development / procurement costs (especially in software-intensive products of Division A.2 and Division A.3)
  - Open source components can be reused within the company decreasing in-house development costs, ensuring software consistency and compatibility
  - Open source software can be a platform of interaction and collaboration between product teams within and across Company A Divisions, resulting in knowledge and resource sharing
  - Potential cooperation with OSS consortia (e.g. OSADL) working on certifying OSS components (e.g. Linux) for the aerospace industry
  - Contributing to OSS communities can improve development efficiency and quality of used OSS components (for generic / non-differentiating components).

Threats of Ungoverned Open Source Use in Products

After analyzing the strengths, weaknesses, and opportunities of open source use in products, we went on to analyze the threats of such use without corporate open source governance. The latter was the case at all the Divisions of Company A.

In our initial situation assessment, we found that Company A extensively used open source components in its products across all of its Divisions we studied. Some Company A Divisions also contributed to open source communities. Both open source use and contribution are beneficial, when they are properly governed and regulated. However, the initial situation analysis at Company A indicated that FLOSS use is not properly governed or regulated. This unregulated FLOSS use and contribution carried significant threats to the company, including financial risks caused by non-compliance to open source licenses and other risks we described in this subsection.

The potential threats included:

- financial costs, such as
  - paying penalties to the copyright holder of the non-compliant open source software
  - paying royalties, license or other fees
  - revenue loss due to product recall / sales stop, etc.
- compliance costs, such as
  - delay of product releases due to unregulated compliance process
  - removing certain open source components from products
  - replacing open source components with commercial or in-house ones
  - hiring lawyers for license audit, etc.
- technical costs, such as
  - disclosing open source components used in products, including information on potential security vulnerabilities
  - ensuring no security vulnerabilities can be used to harm company's products or customers
- other losses, such as
  - loss of intellectual property (e.g. if a company is forced to publish the source code of its own products that use copy-left licensed open source components)

  - loss of reputation with customers and open source communities (thus increased risk of more lawsuits).

As a result of the initial situation assessment of corporate open source governance at Company A, we identified some key characteristics common across all the divisions on the following topics:

- Open Source Use Analysis
- Open Source Contribution Analysis
- General Governance and Inbound Governance
- Outbound Governance
- Supply Chain Management.

We presented the summarized results for each of the above-mentioned topics of corporate open source governance in the following subsections.

Open Source Use Analysis

- FLOSS use
  - In Products – Growing use
  - In Development – Extensive and critical use
  - In R&D – Extensive use
- Forces for more FLOSS use
  - Developers
    * Familiar tools and components
    * Source code availability, less bugs
    * Strong FLOSS capabilities
  - Some middle managers
    * Easier to meet customer requirements

        * Cheaper than commercial software and less effort than in-house development
    - Legal department
        * Limited number of common FLOSS licenses
        * Simpler licenses compared to proprietary software licenses
- Forces against FLOSS use
    - Top management
        * Conservative risk-averse approach, but changing
        * Little awareness of FLOSS benefits (focus on risks)
    - IT departments at Company divisions
        * Traditional preference of commercial enterprise software (e.g. Windows, SAP, etc.)
        * No resources for FLOSS maintenance, updates, etc.
        * Problematic communication and collaboration with product development teams
    - Some middle managers and developers
        * Unclear processes and regulation of FLOSS use
        * Unwanted responsibility for checking FLOSS licenses and deciding on use
        * No warranty and little support for FLOSS
        * Limited FLOSS capabilities and no training
        * No economic evaluation model for FLOSS vs. Buy vs. Make

Open Source Contribution Analysis

- FLOSS contributions
    - Minor bug fixes
    - Undifferentiated new functionality additions
    - No know-how / intellectual property loss
- Forces for more FLOSS contribution

  - Developers
    * Familiar with benefits and risks of contribution to FLOSS projects
    * Didn't want to repatch bug fixes, changes with new releases of FLOSS components
    * Wanted to publicly contribute to FLOSS communities (implied motivation – building reputation as good developers)
  - Some middle managers
    * Saved development resources
    * Wanted to give back to the community and increase company's reputation

- Forces against FLOSS contribution
  - Top management
    * Conservative risk-averse approach, strictly against FLOSS contribution
    * Little awareness of FLOSS contribution benefits (aware of risks)
  - Legal department
    * Unclear rules / strategy of FLOSS contribution
    * Fears potential loss of intellectual property
  - IT departments at Company divisions
    * Unwanted responsibility for maintenance of company's FLOSS projects and contributions
  - Some middle managers and developers
    * Unclear processes and regulation of FLOSS contribution
    * Unwanted responsibility
    * Limited FLOSS capabilities and no training
    * No economic evaluation or rationale for FLOSS contribution.

General Governance and Inbound Governance

- Processes and regulations
  - No defined processes for inbound FLOSS governance in production including component selection, component approval, and component integration
  - Contradictory rules on inbound FLOSS use from different managers, departments
  - No central documentation or repository of FLOSS use, reuse, and related metadata (licenses, versions, etc.)
  - No company strategy, policies or guidelines on FLOSS use
- Roles and responsibilities
  - Unclear, changing, and not unified roles in FLOSS governance
  - Extensive responsibilities put on developers (license checks, understanding of legal consequences, audit and code scans for supplied code, IT work, etc.)
  - No clear decision making body for FLOSS governance
  - No responsible bodies for decision escalation
  - Little division-internal knowledge of FLOSS license compliance (mostly outsourced across Division A.5)
- Component reuse and knowledge sharing
  - FLOSS component and license repositories available only inside individual development teams; not shared with other teams or divisions
  - Knowledge shared within development teams in single locations (mainly through word of mouth)
  - No centralized FLOSS component repository
  - No white/black lists of FLOSS licenses available centrally
- Knowledge about FLOSS governance

  - Developers – High Awareness
  - Legal department – Medium-High Awareness
  - Middle management – Low-Medium Awareness
  - IT department – Low-Medium Awareness
  - Top management – Low Awareness

- FLOSS governance education

  - No trainings / education offered on FLOSS use, governance or compliance

Outbound Governance

- Roles and responsibilities

  - Developers – main role of responsibility for FLOSS license compliance (for familiar licenses)
  - Technical / project managers – follow developers' recommendations
  - Central legal department – rarely involved on division level, on company level – involved in case of license complications (main contact – Legal Counsel in Division A.5)
  - IT department – rarely involved, but had untapped capability of license checking during outbound compliance
  - Top management – involved by setting strategic approach

- Tools

  - License identification and compliance checks were done manually for outbound governance
  - No tools used to scan code or identify FLOSS licenses
  - Few tools to collect and track FLOSS component architecture or license information

- Currently used licenses

  - Permissive licenses (e.g. MIT license) - Yes

  - Semi-permissive licenses (e.g. LGPL) - Yes
  - Copyleft licenses (e.g. GPL) – Very rarely, but happened

Supply Chain Management

- Supply chain (software)
  - Commercial software
  - Open source software
  - Software support and consulting
- Supply chain management
  - Supplier contracts (mainly)
    * No / rare mention of FLOSS license compliance in contracts
    * No requirement for FLOSS bill of materials supplied (no use of SPDX format or other machine readable formats)
  - Supplier audit
    * Developers checked supplied software for functionality requirements only
    * If supplied code went into products, developers checked FLOSS licenses attached, but didn't look into bills of materials
    * Mainly manual checks, limited tooling involved (e.g Maven plug-in for licenses)
    * No surprise audits for supplied code in terms of FLOSS compliance and governance

Initial Situation at Division A.1 – Pilot Project

We chose Division A.1 as our pilot study for theory evaluation, because one of its projects struggled with several issues related to the use of open source components in 2015-2017. After this project, Division A.1 decided to move towards more software intensive markets, thus anticipating open source

use becoming a recurring practice, which needed to be regulated and defined by open source governance processes. In the first phase of our project we assessed the initial situation of open source governance at Division A.1 to identify the governance needs of the division and by extension some of the needs of Company A.

We found that Division A.1 was already using some open source software in products, in software development, and in research and development (R&D). These were distinct use cases in terms of corporate open source governance. Figure 4.2 illustrates a matrix of two major use cases and user groups of FLOSS adoption at Division A.1 we identified during the initial assessment of open source use and governance at Company A.

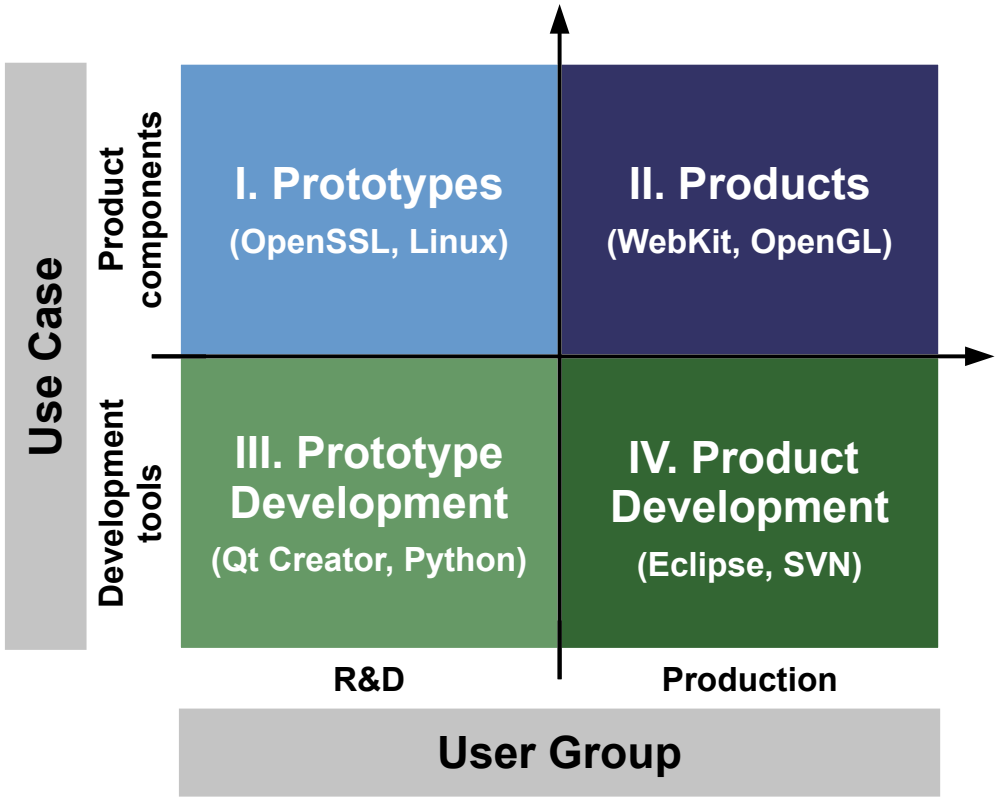


Figure 4.2: Case Study A – Situation Assessment at Division A.1 of Company A: Matrix of Open Source Use

The R&D department was the earliest user of open source software at Division A.1, especially for

development tooling. Some of the open source software used at Division A.1 includes Qt Creator[3], Eclipse[4], Linux[5], Yocto[6], and GCC Compiler[7].

Using open source tools and components for their prototypes, developers were able to demonstrate functional mock-ups to their potential clients without the otherwise high initial investment, such as buying the commercial parts instead of using open source ones. Open source software was deemed more advantageous for the following reasons:

- Commonplace and efficient software for R&D
- Quickly available, transparent and modifiable tools
- Low cost and high quality for tools
- R&D team with existing competences in open source tooling
- No need for rigorous license compliance check for non-commercial / demo use
- OSS communities for support.

One weakness we identified in using open source for R&D was the lack of structured processes for open source knowledge and competence transfer to other teams in Divisions A.1. This resulted in the rough transition from R&D to production due to the limited open source use in production (while R&D prototypes were mainly built using open source components), as well as the initially little attention to open source licenses in prototyping.

The newest use of open source software at Division A.1 was in products that were becoming more software intensive and less specialized over time. Previously the software components were very specialized for the aerospace industry, which meant that product development teams couldn't find suitable open source components to use.

[3]Qt Project – https://www.qt.io
[4]Eclipse IDE – https://www.eclipse.org/ide/
[5]Linux Foundation – https://www.linuxfoundation.org/
[6]Yocto Project – https://www.yoctoproject.org/
[7]GCC, the GNU Compiler Collection – https://gcc.gnu.org/

However, as Division A.1 was entering into new markets that required products with less specialized software, the product development teams had to use more and more open source components (such as web browsers or audio encoders). Developing such components in-house would not be feasible. Instead, the alternative solutions would be either using open source software or commercial software from third-party suppliers. The former was often deemed preferable for the development of non-differentiating features of Divisions A.1's products.

The pioneer project we studied was a software system for managing a plane cabin. In this project, open source components were used for complex and generic functionalities (e.g. browser, audio/video drivers, players, and codecs). Some of the open source software used included WebKit, OpenGL, and Fluendo. Such use had the following advantages:

- Resource saving (generic components were not developed in-house, nor purchased from commercial suppliers)
- Mature open source projects and communities, thus few bugs and easier maintenance
- Customer requirements were met (functionally impossible without open source components for the studied project)

Some of the disadvantages were:

- No structured compliance process (e.g. legal review, license scanning) and associated risks
- Need for in-house preparation of OSS components for special industry certification (e.g. designing requirements for the existing open source components, ensuring requirement-source code traceability) and associated costs

The open source became a priority for Division A.1 after the aforementioned project, which started in 2015 and was ongoing when we were conducting our initial situation assessment as part of Case Study A. Before that, Divisions A.1 had used open source tools only, but only very limited open source software in products. This was partially due to the little need for such use, and partially

due to company policy generally not allowing the use of third-party components (without further definitions or regulations), as well as the overall conservative culture on the issue (especially in the legal department and in the management). Before 2015 most products were highly safety critical and had very specialized software that was not replaceable by open source software. However, after 2015 open source started to become necessary for use in products due to the company's shifting focus on new markets, namely that of software systems outside of the plane cockpit (one the most safety critical areas of a plane). While lacking open source governance, software developers had to include certain open source components in the project that started in 2015. The main reason was its necessity in order to meet the functional requirements the customer had, which among others included integration with Linux-based systems on the customer's end.

We summarized the key characteristics of the initial situation assessment at Division A.1 – Aerospace of Company A below:

- Company
  - Independent division with own functional departments (IT, Legal, etc.)
- Market
  - B2B market with large enterprise customers
  - Market were strictly regulated through standards and certification
- Products
  - Main products were airplane parts and systems with long life spans
  - Products traditionally had little software (built in-house)
  - Market demanded more software as part of products
- FLOSS Use
  - FLOSS use in products (new and limited; no GPL-licensed software) and in development

  - No FLOSS contribution to open source projects
  - Notable FLOSS components used included WebKit, OpenGL (in products)
  - FLOSS capabilities were concentrated in the development team of one specific project

Initial Situation at Division A.2

We summarized the key characteristics of the initial situation assessment at Division A.2 – Internet of Things of Company A below:

- Company
  - Recently separated from Division A.4 (still sharing some business functions, e.g. IT)
- Market
  - B2B market with a multitude of medium-sized enterprise customers
  - Market was not strictly regulated
- Products
  - Main products were smart home and IoT devices, systems, and solutions (middleware)
  - Products highly depended on and included software (FLOSS, commercial, built in-house)
  - Market demanded more software solutions
- FLOSS Use
  - FLOSS use in products (extensive and critical use; also GPL-licensed software) and in development
  - FLOSS contributions happened though rarely (by developers), but were not regulated with formal governance
  - Notable used FLOSS components included Linux (in products)
  - FLOSS capabilities were in a development team with management awareness

Initial Situation at Division A.3

We summarized the key characteristics of the initial situation assessment at Division A.3 – Metering of Company A below:

- Company
  - Independent division with own functional departments (IT, Legal, etc.)
- Market
  - B2B and B2C markets with a multitude of (small and medium size) customers
  - Market was regulated to some extent (depending on country markets)
- Products
  - Main products were water, heat, gas meters, metering systems, and solutions
  - Products traditionally had little software (built in-house), but started including more and more complex software components
  - Market demanded more software solutions, especially focused on IoT and big data
- FLOSS Use
  - FLOSS use in products (extensive and critical; no GPL-licensed software) and in development
  - FLOSS contributions happened rarely (by developers), but were not regulated with formal governance
  - Notable used FLOSS components included Java libraries (in products)
  - FLOSS capabilities were concentrated in the development team, and in the management; IT was somewhat aware of FLOSS governance

Initial Situation at Division A.4

We summarized the key characteristics of the initial situation assessment at Division A.4 – Electronic Assemblies of Company A below:

- Company
  - Independent division with own functional departments (IT, Legal, etc.)
- Market
  - B2B market with a multitude of medium and large-sized customers
  - Market was regulated to some extent (depending on country markets)
- Products
  - Main products were electronic assemblies and system components for washing machines, tumble dryers, dishwashers, stoves, ovens, and refrigerators.
  - Products traditionally had little software (built in-house), but started including more software (e.g. touchscreen user interface (UI) with complex functionalities)
  - Market demanded more software solutions, especially focused on automation and IoT
- FLOSS Use
  - FLOSS use in products (extensive and critical; no GPL-licensed software) and in development
  - No FLOSS contribution to open source projects
  - Notable used FLOSS components included open source UI components (in products)
  - FLOSS capabilities were concentrated in the development team; and in the management; IT was somewhat aware of FLOSS governance

Initial Situation at Division A.5

We summarized the key characteristics of the initial situation assessment at Division A.5 – Information Technology of Company A below:

- Company
  - Internal division providing IT services and development tools to other divisions
  - Collaborated with smaller IT departments in each division
  - Was going through organizational restructuring (towards more centralization from division-specific IT departments to Division A.5)
- Market
  - Internal supplier, no external markets
  - Diverse internal users in all divisions
- Products
  - Development tools for production in all divisions
  - Supporting systems and IT infrastructure (e.g. network, storage) for internal communication
- FLOSS Use
  - Traditionally used little FLOSS, instead procured third-party software components and systems
  - Internal users demanded more FLOSS components and tools
  - Provided all divisions with significant centralized support in FLOSS governance, especially legal support
  - No FLOSS contribution to open source projects
  - FLOSS capabilities were concentrated in the legal team; management was somewhat aware of FLOSS governance.

4.4.2 Evaluation of Getting Started

We started the guided implementation of the getting started section of our handbook at the pilot project at Division A.1 of Company A. This section captured the industry best practices we had identified by analyzing the expert interviews at companies with an advanced understanding of corporate open source governance. The topic of getting started with FLOSS governance was one of the major subtopics of our proposed theory. To test this part of our theory we guided the implementation of the respective handbook section at a production level project (pilot project at Division A.1) at a company with no governance in place (Company A).

After assessing the initial situation of open source governance at Company A and after having developed the getting started part of our theory, we organized a workshop at Company A with our primary contact employees and their colleagues from the pilot project at Division A.1 chosen for the theory evaluation due to its accessibility and urgent need of open source governance practices. The latter was based on the division's recent experience of struggling with the lacking open source governance, as one of the customers had requested a mandatory use of certain open source components (for compatibility reasons). During this workshop we presented the developed handbook section to the stakeholder employees, going into the details of select best practices and workflows that interconnected several practices. We called such workflows process templates, as a company using the handbook would need to adjust and modify the proposed process workflows or create new ones that would fit the company-specific processes and guidelines.

Together with our partners from Company A, we then selected the pilot project and the team that would carry out the implementation of the handbook's getting started section. After the introductory workshop, we met with the pilot project team that consisted of two software developers (from production), an R&D software developer and an R&D manager. During the guided implementation the product software developers focused on applying and using the getting started best practices, while the R&D developer and manager dealt with the overall introduction of the handbook including the application of the inbound governance section we discussed in Section 4.4.3.

The two software developers working on a Division A.1 product and tasked with the implementation of the getting started section of the handbook started by reading the section and asking any questions they would have to us. For example, one of the questions that was raised concerning best practices A.3.9 (*OSGOV-GETSTA-PROANA-3.1. Run open source use analysis in products*) and A.3.10 (*OSGOV-GETSTA-PROANA-3.2. Document current open source use*) was about the specific metadata of the used open source components that needed to be documented. Before following the handbook best practices in running open source use analysis in products and documenting the identified open source components in use, the pilot project team wanted to clarify and document the specific metadata for each open source component. Their initial suggestion after reading the handbook was to use the following metadata:

- license name
- license version
- use case (internal tool, customer application software, delivered operating system, etc.)
- restrictions (modifiability, source code publication, etc.).

This question indicated to us that this part of our theory was not detailed enough, therefore lacking applicability, which corresponded to one of the theory evaluation criteria we had outlined before in the case study protocol, see Appendix E. To address this, we presented further metadata they could consider based on our theory:

- Component ID
- Component name
- Component address / location
- Product / Project ID
- Product version
- Product / Project name

- Multiple licenses (y/n)
- Copyright holder(s)
- Linkage type to the rest of the (software) product (e.g. dynamic or static)
- Has the component (with its unchanged license and version) been used in the company before (can be automatically identified); if already used (a reference to the previous use).

The pilot project team added the above-mentioned metadata to their initial suggestion of identifying the use case and the usage restrictions of an open source component. They then requested the defined metadata from the developers involved in the pilot project, following the handbook best practice A.3.2 (*OSGOV-GETSTA-PROANA-1.1. Use one mandatory survey for initial assessment*). Following the best practices A.3.1 (*OSGOV-GETSTA-PROANA-1. Use a combination of methods for product analysis*) and A.3.3 (*OSGOV-GETSTA-PROANA-1.2. Establish a process of continuous reporting and assessment*), the pilot project team went on to analyze more of the used open source components by scanning several products and starting the establishment of a process of continuous reporting and assessment for future open source component additions. This resulted in the first automated scan at Company A using an open source tool for FLOSS governance, called FOSSology[8] [58], following the best practice A.3.4 (*OSGOV-GETSTA-PROANA-1.3. Select and use governance tools for automation*). The tools were chosen temporarily for the getting started process, as it did not require a lengthy procurement process necessary for the proprietary tooling alternative. However, the pilot project team was explicit that further tool comparisons would have to be performed before choosing the right long-term tooling of open source governance and compliance used across the company. Running an initial FOSSology scan was aimed at identifying the used but undocumented open source components in the current products at Division A.1 of Company A, their licenses, copyright notices, and other metadata. As a result, the first handbook implementation artifact was created – the FOSSology report with the identified open source components, their licenses, and other metadata. One of the employees (a manager from the R&D department at Division A.1)

[8]FLOSS Governance and Compliance tool FOSSology – https://www.fossology.org/

tasked with implementing the getting started section of our handbook created this artifact, analyzed the results and started the manual review of the identified open source components in the existing product under review. The company-sensitive data has been anonymized. Some of the identified components and their licenses were masked. See the first page of this report in Figure 4.3. See the rest of the report in Section F.2 in Appendix F.

FOSSology

OSS Component Clearing Report [Excerpt] at Division A.1, Company A		
Clearing Information	**Department**	FOSSology Generation
	Prepared by	Employee X (employee_x)
	Reviewed by (opt.)	NA
	Report release date	2018/12/18
Component Information	**Community**	NA
	Component	NA
	Version	NA
	Component hash (SHA-1)	D2D346D1B90E4E1E0F7DB22CD92B6649DA7EF1C2
	Release date	NA
	Main license(s)	Main License(s) Not selected.
	Other license(s)	License(s) Not Identified.
	Fossology Upload/Package Link	http://nas02fra:8081/repo/?mod=showjobs&upload=5
	SW360 Portal Link	NA
	Result of License Scan	0BSD, AFL-2.0, AFL-2.1, [redacted], AMD, ATT, Apache, Apache-1.0, Apache-2.0, Artistic-1.0, Artistic-1.0-Perl, Artistic-2.0, Autoconf-exception, BSD, BSD-2-Clause, BSD-2-Clause-FreeBSD, BSD-2-Clause-NetBSD, BSD-3-Clause, BSD-4-Clause, BSD-4-Clause-UC, BSD-possibility, BSD-style, BSL-1.0, Bison-exception, Bison-exception-2.2, CC-BY-NC-SA-3.0, CC-BY-ND-2.0, CC-BY-SA, CC-BY-SA-3.0, CC0-1.0, CMU, CNRI-Python, ClArtistic, Cryptogams, DOC, Dual-license, FSF, FTL, Freeware, [redacted]

Figure 4.3: Case Study A – First Page of FOSSology Report Excerpt from Division A.1

The FOSSology report illustrated the large number of the previously unidentified open source components that were already part of Company A products. More than 10000 open source components (with different copyright notices) were identified in this first small scale scan. FOSSology also identified about 100 open source licenses (each license version and variation was counted as a different license as is the industry best practice captured in our theory). Such numbers after the small scale scan at Division A.1 came as a surprise to the pilot project team, but also confirmed the high relevance of our theory (and open source governance in general) for the company.

The pilot project team then started the manual review of the open source components and their metadata identified in the scan, adding the cleared components and their metadata into a repository that was created to track the use of open source at Company A. At this early stage, this repository was a spreadsheet with all the components, their locations, licenses, and other metadata, which fell short from the industry best practice A.3.6 (*OSGOV-GETSTA-PROANA-2.1. Create product architecture model*) from our theory. However, this was a common practice (observed during theory building) of starting with a spreadsheet and transitioning into a more complex product architecture model over time. Running a product analysis and setting up an early component repository, the pilot project team addressed one of the issues we identified in the initial situation assessment at Company A as a whole, and at Division A.1 in particular – open source components were used, but such use was not documented or governed, which carried compliance risks for the company. Having identified the used open source, the pilot project started the compliance checking process for the components that were already in use. As ways to mitigate the identified risks related to the use of open source, our colleagues expected to follow some of the best practices from the getting started section of our handbook, namely practices A.4.7 (*OSGOV-GETSTA-IPRISK-3.1. Replace problematic components*) and A.4.8 (*OSGOV-GETSTA-IPRISK-3.2. Decouple problematic components*).

After observing the guided implementation of the getting started section of the governance handbook based on our theory, we went on to evaluate the transferability of the proposed theory's part on *getting started with corporate open source governance*. We used the following evaluation criteria

defined in the evaluation case study protocol, which we presented in Appendix E:

- *Completeness*
- *Variability*
- *Structure*
- *Comprehension*
- *Understandability*
- *Applicability*
- *Relevance*
- *Significance*
- *Usefulness.*

As planned, we evaluated the implementation of the getting started section of the open source handbook as a proxy for the evaluation of the getting started part of the proposed theory. Our theory evaluation was based on matching the patterns that emerged from the handbook implementation with the patterns proposed in the getting started part of our theory. We analyzed how the actual handbook implementation differed from the proposed theory, reporting the results in the context of each of the evaluation criteria.

Completeness was assessed for the getting started section as a whole, as it evaluated whether the section had an adequate beginning, middle, and end, as well as whether it lacked any practices the company needed when applying the handbook. The employees tasked with the implementation of the getting started part of our theory reported in our follow-up interviews that the handbook section on the topic had an adequate level of completeness without any significant gaps or unanswered questions they encountered. However, during our direct observation, we noted that the pilot project faced some completeness related issues, which mainly related to Division A.1 specific processes. For example, the development process at the division provided checklists for software developers to fulfill before moving into the next cycle of the development process. Our theory did not

take into account this specific need of Company A (as our theory was developed based on industry best practices at expert companies who did not use such checklists). To complete this gap, the R&D developer tasked with the handbook adoption at Company A planned to add some practices to the getting started section before the company-wide roll-out. We observed another challenge associated with such an extension – the lack of the handbook versioning for future extensions. This challenge was caused by the Word and PDF formats of the handbook (required by the Company A – FAU contract we had to conform to). Versioning of the handbook extensions at Divisions A.1 and possibly in other divisions would create a significant effort of syncing and merging all the changes ensuring that there was one authoritative version of the handbook at all times, which would, in turn, ensure the consistency of open source governance processes. Having thought about this issue in advance (during theory building), we had collaborated with a local start-up – Editive (previously Sweble)[9] to offer a collaborative tool that would help with managing multiple versions of the governance handbook, as well as their syncing and merging. Editive's software as a service (SaaS) solution, based on the research by Dohrn and Riehle [41], would enable its users (in this case Company A employees) to maintain one up-to-date version of the handbook, while allowing individual divisions to modify it with division-specific practices or changes. Additionally, the tool would enable the responsible employees (e.g. the to be established Open Source Program Office) to merge the modifications from the division versions (if such changes could benefit others, too). Furthermore, Editive would enable an easy distribution and notification about any updates or additions to the corporate open source governance handbook at scale. We had presented this solution at Company A during our case study creating several illustrative examples, see the Editive screenshot in Figure 4.4.

[9]Editive Start-up (previously Sweble) – https://editive.com/

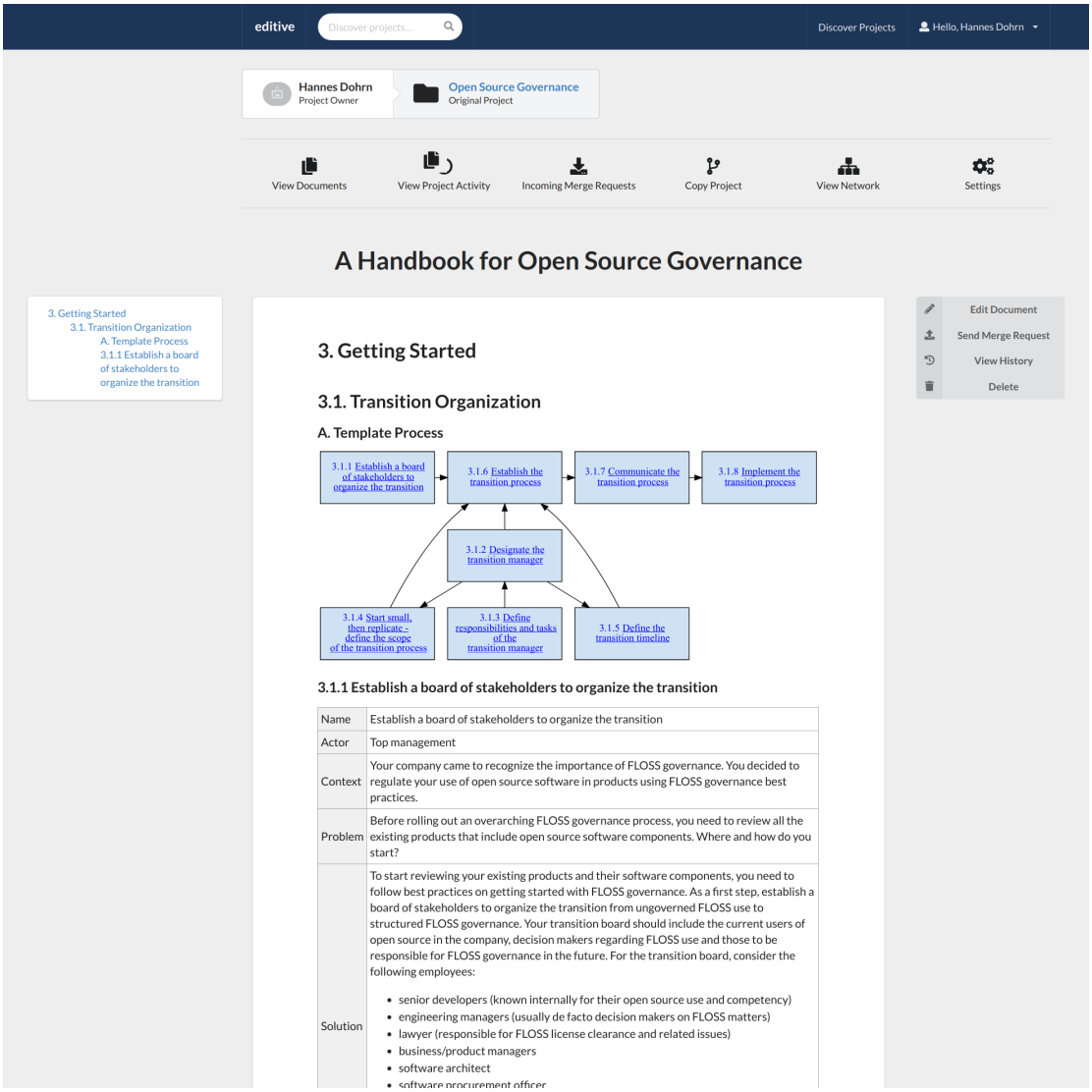


Figure 4.4: Case Study A – Screenshot of an Excerpt from the Getting Started Section of the Governance Handbook – Editive Collaboration Platform

However, though such a tool would have eliminated the completeness and versioning related limitations of the Word or PDF formats, this solution was not adopted by Company A during Case Study A as the negotiations between Company A and Editive were still ongoing.

Variability was also assessed for the getting started section as a whole (similar to completeness), as it evaluated whether the section had a balanced mixture of concepts for getting started with cor-

porate open source governance and not overly focused on a single concept. During Case Study A, we observed that the balanced design of this part of the proposed theory translated into an equal coverage of different getting started concepts, such as transition management, product analysis, and IP-at-risk analysis. This observation was also confirmed by the employees implementing the handbook at Company A, who highlighted that no concept was singled out and presented in more detail than others. Such an assessment led us to the positive evaluation of the variability criteria of this part of the tested theory.

Structure was assessed for both the getting started section and for the individual industry best practices from the proposed theory, as we evaluated how well-structured both the section and the individual practices were. For the section as a whole, we evaluated whether its different parts were structured in a logical and interconnected manner. The pilot project employees who were implementing the handbook at Company A appreciated the interconnecting links between individual best practices within the getting started section (this was also the case for the sections), as such links created workflows that could be made into company processes and were already ingrained into the theory, therefore, making it easier to apply at the company. Using such links between the practices for the section, the R&D developer created a structured overview of the getting started part of our theory presented together with the inbound governance overview in Figure 4.5 of Section 4.4.3. We did observe an issue with the links between the best practices in an early deliverable of the handbook section to Company A. However, this was a technical issue and was fixed once the pilot project team informed us about the broken links issue in the PDF version of the handbook. As to the structure of the individual best practices, all of the employees involved in handbook implementation at Company A noted the value of using the structured presentation format for the industry best practices from our theory – the Context-Problem-Solution pattern format that made the practices more digestible.

Comprehension was assessed for both the getting started section and for the individual industry best practices from the proposed theory, as we evaluated how well the theory answered the problems

companies with little to no governance would have, as well as whether the proposed best practices went into enough detail on their respective issues. Evaluating the section as a whole, we found that some of the workflows or process templates made of several best practices were confusing to the users of the handbook in the pilot project team. Moreover, we identified that some of the workflows that were giving an overview of the getting started section (among other aspects of the theory) did not comprehensively capture all the interlinked best practices in the section. To address this (together with the above-mentioned issue related to the structure of the section) Company A put together a comprehensive overview of the section to be used in the company-wide implementation of the handbook after the pilot project, which we presented in Figure 4.5 of Section 4.4.3. The same technique was later applied to all the sections of the handbook by the R&D developer of Company A. See all the handbook implementation artifacts with such overviews in Section F.1 of Appendix F. Furthermore, several employees noted that the handbook section was too general and not customized enough for Company A, therefore lacking essential details for the immediate implementation at the company. We recognized this issue during our direct observation, too. However, this was a natural limitation of our theory, as it was built based on the data from expert companies, which was not specific to Company A. Moreover, our goal in theory building was not creating a theory that would apply perfectly to one or two companies of our choice, but rather abstract from industry best practices creating a theory that can be applied widely. In other words, we aimed at transferability, whose evaluation we presented in this chapter. As to the evaluation of the specific best practices, some lacked detail and were not comprehensive according to the pilot project team. For example, they mentioned that the best practices A.1.4 (*OSGOV-GETSTA-TRAORG-4. Start small, then replicate*) required more details before it could be implemented. Namely, it was unclear what was meant by "replicate" in this context. We guided the team in explaining that in this context it meant transitioning towards open source governance in a select project at a company, then scaling up this transition to other teams across the company. After this clarification, the issue with the comprehension of this practice was resolved, though some of the employees asked for further details on

the specifics of choosing the right scope for getting started with open source governance.

Understandability was assessed for both the getting started section and for the individual industry best practices from the proposed theory, as we evaluated how understandable the theory and its representation in the handbook format were to the employees implementing and using the handbook. We focused on assessing the understandability of both the intentions and the specifics of the proposed theory. Evaluating the section as a whole, we found that the pilot project employees had to read the section carefully, attentively, and completely to ensure the full understanding of the section. This costed a significant amount of time (in average two months per employee) given that implementing the governance handbook was not a full time task for the pilot project employees the Company A. The pilot project team recognized that the users of the handbook (e.g. developers, middle managers) would not read the getting started section in full, which could potentially lead to understandability challenges. To prevent such issues, the pilot project team set out to integrate the getting started section of the handbook into the existing software development process at the company. The R&D employees (a developer and a manager) tasked with the adoption and roll-out of the handbook at the company mentioned the potential understandability issues as one of the motivations for creating a company-internal guideline that would spell out the highlights of open source governance and how they fit Company A's existing processes. Our handbook would be attached to this guideline to increase the understandability of the specific best practices for the stakeholder employees, while not forcing all the employees to read the handbook in full. The latter was deemed unrealistic at Company A, which constituted a key finding of our theory evaluation. Furthermore, the pilot project team planned to create employee training (for new hires and old employees), e-books, and other educational materials covering the highlights from the getting started section of the handbook in an easily digestible and understandable way. As to the select best practices, their evaluation demonstrated that some of the handbook users had understandability issues caused by the language used to present the theory. For example, the best practice A.3.2 (*OSGOV-GETSTA-PROANA-1.1. Use one mandatory survey for initial assessment*) had an unclear definition of what was meant by

”survey” in this context, which was clarified during the guided implementation. The implementation pattern (affected by the initial understandability issue) showed that the pilot project confused the survey for an initial assessment with the continuous reporting from the proposed handbook pattern – the best practice A.3.3. We cleared this confusion by addressing the potential overlap between the two industry best practices. We highlighted that the pattern suggested by our theory proposed the parallel application of these practices when getting started with governance. Another example of the understandability issues caused by the language was observed in the implementation of the best practice A.1.4, namely about the unclear meaning of what ”replicate” meant here. We already reported this issue in detail in the discussion of the comprehension criteria of our theory evaluation.

Applicability was assessed for both the getting started section and for the individual industry best practices from the proposed theory, as we evaluated how well our theory could be applied to a company with a different context from that at the expert companies involved in theory building. We evaluated how generalizable the getting started part of the theory was, as well as how much the evaluated best practices needed to be adjusted to become applicable at Company A. Evaluating the section as a whole, we found that the biggest challenge for the applicability was the lack of a customized process (for Company A) of the getting started with corporate open source governance at Company A in particular. By design, our theory presented only general industry best practices on the topic, not customizing them for a ready implementation at one specific company. This led to an initial mismatch of the expectations between the researchers from FAU (the author of this dissertation and colleagues supporting the project) and the pilot project team at Company A (from Division A.1). After referring our colleagues from Division A.1 to the statement of work signed between FAU and Company A, the misunderstanding was cleared, but the applicability issues remained. To address them, the pilot project team tasked the R&D employees (a developer and a manager) to create a Company A specific customized set of processes and practices for open source governance based on our provided handbook. Observing such application of the handbook, we recognized that it helped Company A apply the key concepts from our theory to their context through this interface.

The pilot project defined a company-internal guideline that was used to integrate and apply the proposed industry best practices at Company A in a predictable and manageable manner. One of the techniques used in this guideline was the definition of software development checklists (commonly used at Company A) based on the getting started best practices, such as best practices A.3.9 (*OSGOV-GETSTA-PROANA-3.1. Run open source use analysis in products*) and A.4.9 (*OSGOV-GETSTA-IPRISK-3.3. Require bill of materials for supplied code by 3rd party post-factum*). Such checklists would eventually become mandatory parts of the software development process, and their complement would allow the developers to pass through development process gates (commonly used at Company A). Furthermore, evaluating the applicability of individual best practices at Company A, we observed that some practices did not apply at Company A, being out of scope for the planned getting started process there, such as the best practice A.4.11 (*OSGOV-GETSTA-IPRISK-4. Analyze the security risk of using an open source component*). This best practice was out of scope for the pilot project as security-related issues were handled mainly by the IT departments at Company A, and were not involved in the governance pilot project, though their involvement into governance efforts at the company was planned for the future company-wide roll-out (of corporate open source governance processes).

Relevance was assessed for both the getting started section and for the individual industry best practices from the proposed theory, as we evaluated how relevant the theory was to Company A (and, by proxy, to its employees) in terms of addressed the company's needs of getting started with the corporate open source governance. Evaluating the section as a whole, we found that the employees in the middle management of Division A.1 clearly recognized their needs for getting started with governance, which was addressed by the proposed handbook section. Further confirming the relevance of the proposed theory we observed that company lawyers, developers, and technical managers found that the handbook section answered their questions around open source governance, clarifying the key concepts and providing actionable advice of dealing with challenges of getting started with governance. The getting started section was of special interest to Company A, as it

did not have any formal open source governance in place at the time of the handbook introduction. This meant that the pilot project needed guidance with introductory best practices – a need, which was addressed in the handbook section under evaluation. Company A employees also referred to the potential risks of the ungoverned open source use (also captured and presented in the initial situation assessment earlier in this chapter) matching these risks to the relevant solutions from our theory. For example, one observation from the initial situation assessment was on the lack of open source license interpretation across Company A. Having and using such license interpretation would be of high relevance for any company getting started with FLOSS governance. This was also the case for Company A, whose employees in the pilot project followed our theory's best practices A.4.2 (*OSGOV-GETSTA-IPRISK-1.1. Develop standard license interpretation*) and A.4.3 (*OSGOV-GETSTA-IPRISK-1.2. Use standard license interpretation*).

Significance was assessed for both the getting started section and for the individual industry best practices from the proposed theory, as we evaluated the level of impact our theory had on Company A. Evaluating the section as a whole, we could not fully assess how significant the impact of our proposed theory would be to Company A after the full roll-out across the whole company. As to the pilot project, we recognized that the previous efforts at the company could not address all the needs for governance, while getting started section of our handbook provided significant guidance and support for the transition towards governance in the scope of the pilot project. In this limited evaluation of the theory significance, we observed some challenges, such as the lack of examples in the section. The getting started section did not include examples as we aimed at abstracting from individual examples we had studied from expert companies during theory building. However, we did recognize that adding such examples could be useful in assessing the significance of individual best practices when implementing and using the handbook. Such examples could also help demonstrate the effects (and their significance) of the proposed best practices on the company using the handbook. Though we could not evaluate the significance of the complete getting started part of our theory, we noted that the most significant aspect of our theory to the pilot project employees

was that they could use our handbook as a strong argument in front of the top management of the company demonstrating the significance of corporate open source governance at Company A.

Usefulness was assessed for both the getting started section and for the individual industry best practices from the proposed theory, as we evaluated how much value it added to Company A in solving the key issues of getting started with FLOSS governance, as well as whether it enhanced the employee knowledge on these issues and their solutions. Similar to the evaluation criteria of significance, we could not assess the usefulness of our theory to the whole company during Case Study A, as the handbook was implemented in the scope of the pilot project and the full roll-out was still pending. Evaluating the usefulness of the getting started section as a whole in the scope of the pilot project, we found that the main issue making the handbook less usefulness was its abstract nature. As a result (as already mentioned), Company A had to customize the proposed processes from the handbook section to fit their own internal processes. Not having a "plug and play" handbook was perceived as a not useful without such customization. We agreed with this observation, but highlighted that the handbook was abstract by design, which had been confirmed with Company A in the statement of work between FAU and the company. Nonetheless, we considered this as a limitation to the usefulness of our theory for the companies unwilling to perform the required customization of the handbook. As to the evaluation of the individual best practices, we found that some best practices were not usefulness to Company A, as they were too complex for the use cases of the company. An example was the best practice A.1.1 (*OSGOV-GETSTA-TRAORG-1. Establish a board of stakeholders to organize the transition*), which proposed setting up a board that would include several employees (e.g. developers, lawyers, managers) tasked with overseeing and coordinating the transition towards FLOSS governance. However, given the small size of the pilot project, at the early stage of handbook implementation, Company A considered this practice not to be useful for their context. Instead, the implementation pattern merged the above-mentioned best practice with another one – A.1.2 (*OSGOV-GETSTA-TRAORG-2. Designate the transition manager*) as a more useful solution.

This concluded the evaluation of the getting started part of our theory at Company A, which was the most comprehensive and long-term evaluation study at Case Study A and among all the case studies. However, we went on to evaluate another part of our theory at Company A – Inbound Governance, whose evaluation we reported in the following subsection.

4.4.3 Evaluation of Inbound Governance

In parallel to the implementation and evaluation of the getting started part of our theory at Company A, we started the delivered the inbound governance section of the handbook to the pilot project team at Division A.1. This section captured the industry best practices for the inbound open source governance we had identified by analyzing the expert interviews at companies with an advanced understanding of corporate open source governance. The topic of inbound governance was one of the major subtopics of our proposed theory, and of special interest to Company A, which had started using (on the small scale of the pilot project) the handbook section on getting started, but lacked the next steps that would help establish the processes for the day to day use of open source software in products. To test this part of our theory we guided the implementation of the respective handbook section at a production level project (pilot project at Division A.1) at a company with no governance in place, in the same setup as was the case with the evaluation of the getting started part of the theory.

While the evaluation of the getting started section was ongoing, we delivered the inbound governance section of the handbook to the pilot project team and organized a meeting to go over the details and kick off the guided implementation. The focal topics of inbound governance for Company A were:

- Open Source Component Approval
- Open Source Component Repository and Reuse.

During this meeting we presented the developed handbook section to the stakeholder employees, going into the details of select best practices and workflows that interconnected several practices. We answered the early questions on the content of the section, as well as on its implementation. We then left the pilot project team alone giving them a chance to carefully read the section and start the implementation. In contrast to the implementation of the getting started section that was led by the R&D developer from the pilot project, the implementation of the inbound governance section

was mainly conducted by the R&D manager who was in contact with production teams at Division A.1 who would be the main users of the inbound governance processes at the company. During this implementation, the R&D manager was supported by the R&D developer, and product team developers from the pilot project team.

R&D manager and the developers started by reading the section and identifying the best practices and workflows consisting of several best practices that were of the most relevance and highest priority for the company. Such practices were implemented early on, thus influencing our evaluation of the inbound governance part of the proposed theory. For example, the pilot project team decided that setting up a component repository would be one of the first things to implement following the handbook best practice 3.11 (*OSGOV-INBGOV-COMREU-5. Establish component reuse process*), and best practice *OSGOV-INBGOV-COMREU-8. Create component repository*[10]. The open source components in the repository would also store the governance relevant metadata using the same (metadata) properties identified during the implementation of the getting started section, including:

- Component ID
- Component name
- Component address / location
- Product / Project ID
- Product version
- Product / Project name
- Multiple licenses (y/n)
- Copyright holder(s)
- Linkage type to the rest of the (software) product (e.g. dynamic or static)

[10] *Note*: We did not include the full handbook section on Inbound Governance in the Appendix of this dissertation due to space constraints unlike the excerpts from the sections on Getting Started (presented in Appendix A) and that on Supply Chain Management (presented in Appendix B). Refer to our publication on component reuse for this specific best practice [71].

- Has the component (with its unchanged license and version) been used in the company before (can be automatically identified); if already used (a reference to the previous use).

However, Company A's implementation pattern for the component repository deviated from the proposed industry best practices. Namely, while our theory recommended setting up the repository in a single central location in the company in the best practice *OSGOV-INBGOV-COMREU-13. Provide component repository a single well-defined location*, the pilot project started by a local implementation of the repository with a plan to scale up the repository in the future. Following another best practice from our theory – *OSGOV-INBGOV-COMREU-12. Use tools to create, update and maintain component repository*, the pilot project team reviewed several tooling options and chose the open source software SW360, which could be easily integrated with another tool selected during the implementation of the getting started section – FOSSology. SW360 provided both a web application and a repository to collect, organize and make available information about software components (including open source ones). According to the Eclipse project's official website[11], SW360 established a central hub for software components in an organization, allowing for:

- tracking components used by a project/product
- assessing security vulnerabilities
- maintaining license obligations
- enforcing policies
- generating legal documents
- integrating with other governance tools.

At the time of our evaluation, the pilot project was in the process of establishing and using the component repository locally. They were also collaborating with the IT department for the support with tooling installation at a larger scale.

[11]Component Management Tool Eclipse SW360 – https://projects.eclipse.org/proposals/sw360

Another implementation pattern that did not match the proposed theory was from another subsection of the inbound governance section of the handbook – Component Approval. While our theory recommended defining clear rules for component approval in the best practice *OSGOV-INBGOV-COMAPP-4. Define transparent rules for open source component approval*, the pilot project team decided to review the open source component requests on a case by case basis early on, which would allow them to create and document precedents to be used by the developers in the future. One of the reasons for this decision was the lacking legal competence at Company A, as the main responsibility was put on one legal counsel at Division A.5, who had recently changed his position in the company and was phasing out his open source related responsibilities.

In addition to implementing the specific best practices proposed in our theory, Company A also tasked one pilot project employee with creating a company-internal guideline for rolling out the open source governance handbook as a whole at Company A, which was mentioned in the getting started evaluation. In the course of his preparation for this roll-out, the employee used our handbook to create overview workflows that connected all the proposed industry best practices from our theory. This resulted in another implementation artifact with the goal of visually illustrating and communicating all the processes of corporate open source governance with the stakeholder employees. After reading the handbook, the R&D employee created such an overview workflow and suggested potential ways of implementing the proposed best practices and processes at Division A.1 at first, and then at the whole Company A. He created this artifact going beyond the original content of the handbook, expanding the suggested processes by adding company-specific IDs, color coding, and links between the best practices. Figure 4.5 covered the company processes for getting started with open source governance. Figure 4.6 covered the company processes for inbound governance (focused on component approval). We presented the rest of the Company A's diagrams for the corporate FLOSS governance processes (including on component reuse as part of inbound governance in Section F.1 of Appendix F.

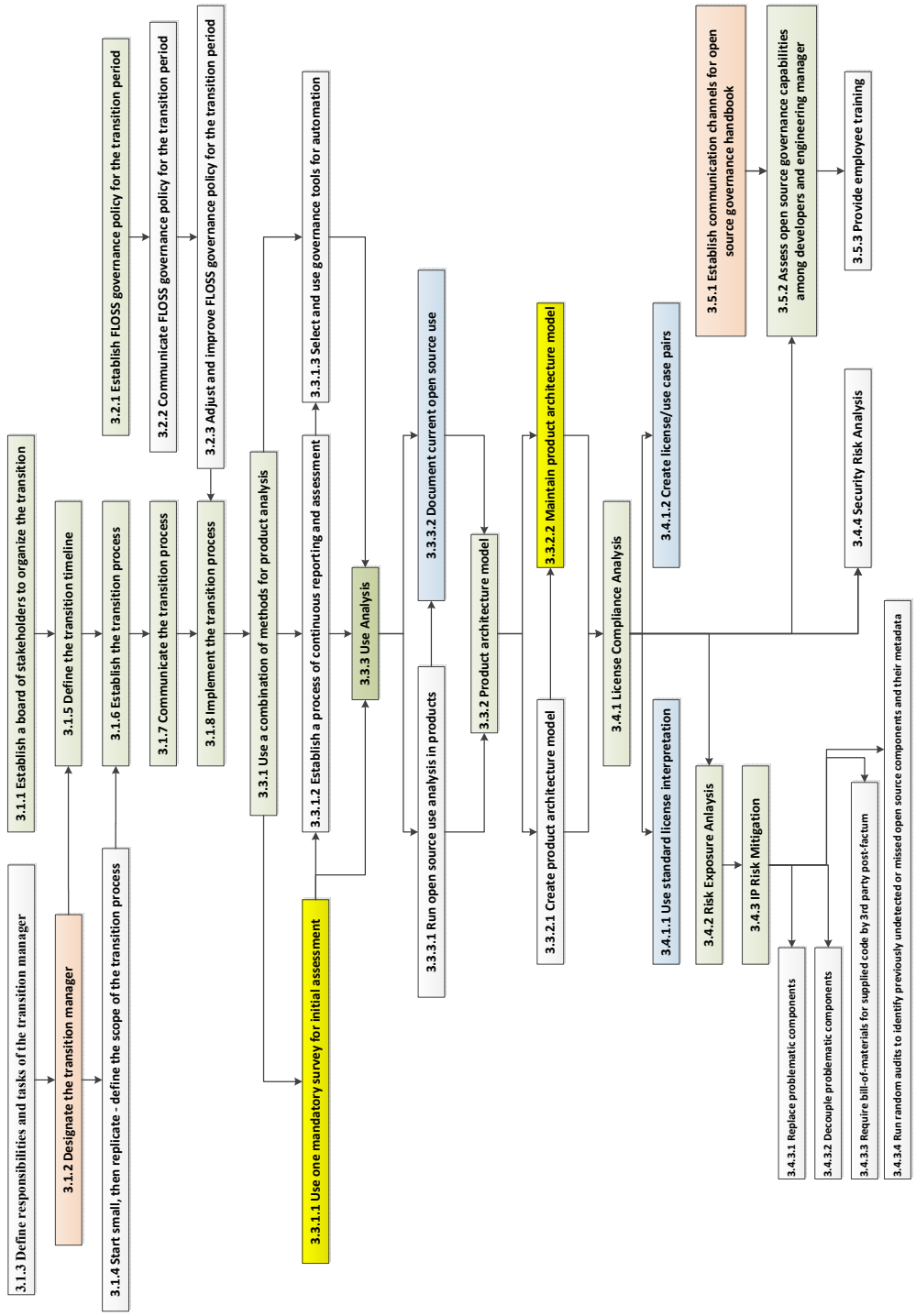


Figure 4.5: Case Study A – Overview of FLOSS Governance Processes on Getting Started

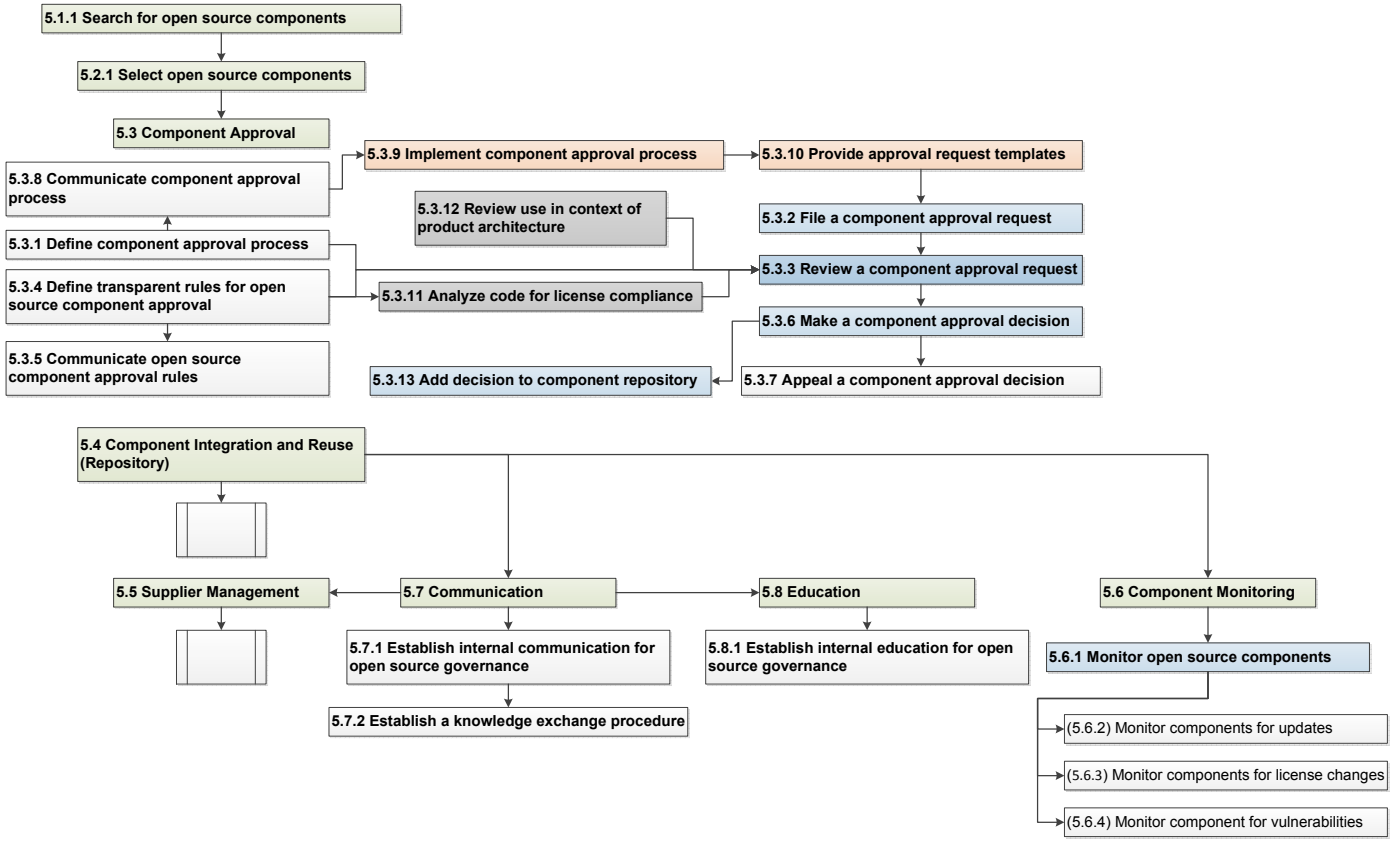


Figure 4.6: Case Study A – Overview of FLOSS Governance Processes on Inbound Governance, focused on Component Approval

After observing the guided implementation of the inbound governance section of the handbook based on our theory, we went on to evaluate the transferability of the proposed theory's part on *inbound open source governance*. We used the same evaluation criteria as for the getting started evaluation, defined in the evaluation case study protocol, which we presented in Appendix E.

Completeness was assessed for the inbound governance section as a whole, as it evaluated whether the section had an adequate beginning, middle, and end, as well as whether it lacked any practices the company needed when applying the handbook. Similar to the getting started evaluation, the employees tasked with the implementation of the inbound governance part of our theory reported in our follow-up interviews that the handbook section's coverage of the component approval and

component repository aspects of inbound governance had an adequate level of completeness without any significant gaps or unanswered questions they encountered. The pilot project team had completeness issues with other subsections of the section, namely that on the component search and component integration. These topics of open source governance were identified and presented as part of our theory, however, we did not present detailed best practices for the topics in contrast to the complete subsections on component approval and component repository. The reason for this was the limited data from the expert interviews one these specific topics. We did recognize this to be a limitation of our theory, however, with a limited scope of this dissertation project, we did not expect to fully cover all the aspects of open source governance. Instead, we had focused on several focal topics, which we presented in the resulting theory in Chapter 3.

Variability was also assessed for the inbound governance section as a whole, as it evaluated whether the section had a balanced mixture of concepts for inbound governance with corporate open source governance and not overly focused on a single concept. We observed that this part of the proposed theory covered most of the key aspects of inbound governance, including:

- Component Search
- Component Selection
- Component Approval
- Component Repository and Reuse
- Component Integration
- Component Monitoring
- Communication
- Education.

After reading the handbook section, the pilot project employees agreed that the above-mentioned list of subsections ensured the coverage of all the key aspects of inbound governance. However, a

limitation to variability was within the subsections. As discussed in the evaluation of the completeness and in Chapter 3 with the proposed theory, we went into full details for two of these subtopics of inbound governance, namely Component Approval and Component Repository and Reuse. As a result, we had higher variability within the subsections on component approval and component repository, while lacking variability within the other subsections that were out of the scope of our research during theory building.

Structure was assessed for both the inbound governance section and for the individual industry best practices from the proposed theory, as we evaluated how well-structured both the section and the individual practices were. For the section as a whole, we evaluated whether its different parts were structured in a logical and interconnected manner. The pilot project employees who were implementing the handbook at Company A appreciated the interconnecting links between individual best practices within and across the subsections on inbound governance. Such links created workflows that could be made into company processes and were already ingrained into the theory, therefore, making it easier to apply at the company. Using such links between the practices for the section, the R&D developer created a structured overview of the inbound governance part of our theory in Figure 4.6. We observed, for example, that the pilot project team followed the link between the best practices (from two different subsections) *OSGOV-INBGOV-COMAPP-13. Add decision to component repository*, *OSGOV-INBGOV-COMREU-8. Create component repository*, and *OSGOV-INBGOV-COMREU-14. Track prior approval data for reuse*. The employees implementing the handbook found the structure of these best practice links logical and integrated them into the company-internal guidelines that would enable the company-wide open source governance. As to the critical aspects of the evaluation, similar to the links between the best practices in the getting started section, we did observe an issue with the broken cross-links in an early deliverable of the handbook section to Company A, which was fixed upon request by the pilot project. As to the structure of the individual best practices, similar to the getting started evaluation, Company A employees involved in handbook implementation noted the value of using the structured presentation

format for the industry best practices from our theory – the Context-Problem-Solution pattern format that made the practices more digestible.

Comprehension was assessed for both the inbound governance section and for the individual industry best practices from the proposed theory, as we evaluated how well the theory answered the problems companies with little to no governance would have, as well as whether the proposed best practices went into enough detail on their respective issues. Evaluating the section as a whole, we found that some of the workflows or process templates did not include all the best practices from all the subsections of inbound governance handbook section. The pilot project team considered this to be a limitation to the comprehension of the section. During the implementation, we explained that no process template included all the best practices as was the design of our theory. Such process templates aimed to aid the customization and the implementation of the handbook at companies, but did not provide comprehensive overviews of all the practices. As a result, the pilot project team developed their own overview of all the workflows for inbound governance. An example focused on component approval was presented in Figure 4.6.

As to the evaluation of the specific best practices, some lacked detail and were not comprehensive according to the pilot project team, confirming a similar finding we had during the getting started evaluation). For example, they mentioned that the best practice *OSGOV-INBGOV-COMREU-16. Search component repository for reusable components* did not comprehensively present how to set up the search mechanism for the component repository. To address this, the pilot project looked into potential tools and the search mechanism available in them.

Understandability was assessed for both the inbound governance section and for the individual industry best practices from the proposed theory, as we evaluated how understandable the theory and its representation in the handbook format were to the employees implementing and using the handbook. We focused on assessing the understandability of both the intentions and the specifics of the proposed theory. Similar to the getting started section evaluation, evaluating the section as a whole, we found that the pilot project employees had to read the section carefully, attentively, and

completely to ensure the full understanding of the section. This took a significant amount of work time. The pilot project team recognized that the users of the handbook would not read the inbound governance section in full, which could potentially lead to understandability challenges. To prevent such issues, the pilot project team set out to integrate the component approval and component reuse processes from the inbound governance section into the existing software development process at the company. At the time of the evaluation at Company A, these processes were not yet applied company-wide. Therefore, we couldn't yet evaluate if the adoption of the inbound governance processes would be understandable to the employees. However, to prevent such issues the pilot project team attached our handbook in full to the company-internal process guideline for open source governance. As to the select best practices, they were clear to the pilot project team and did not have any language related issues. Unlike in this section, in the getting started section there were several language issues.

Applicability was assessed for both the inbound governance section and for the individual industry best practices from the proposed theory, as we evaluated how well our theory could be applied to a company with a different context from that at the expert companies involved in theory building. We evaluated how generalizable the inbound governance part of the theory was, as well as how much the evaluated best practices needed to be adjusted to become applicable at Company A. Evaluating the section as a whole, we found that the biggest challenge for the applicability was the lack of customized processes for inbound governance at Company A. However, similar to the evaluation of the getting started section, this was by design of our theory presentation. To address the issue, the pilot project team created a Company A specific customized set of processes and practices for inbound open source governance, mainly focused on component approval and reuse, based on the handbook we provided. Observing such application of the handbook, we recognized that it helped Company A apply the key concepts from our theory to their context through this interface. As to the evaluation of the specific best practices, we found that some industry best practices cannot be applied at Company A. For example, the practice *OSGOV-INBGOV-COMREU-4. Designate a*

role of responsibility for the component repository, in multiple places in the company could not be applied at the company due to the lack of human resources and the perceived lack of need for such a role. Instead, Company A decided to potentially task some employees in each division to take care of all the issues around open source governance (and not specific aspects of FLOSS governance, as was proposed by our theory). Another issue of applicability at Company A was focused on Division A.1 that had specific software development needs compared to the other divisions, ad Division A.1 had products (in the aerospace industry) with a high level of criticality. Such products required conformance to certain standards and regulations, which influenced some of the aspects of inbound open source governance, namely open source component reuse. As an example, before reusing a component from another product, the developer would have to reverse engineer software development requirements for the used open source component, if the project s/he was working on had a higher level of criticality than that at the project the open source component had been used first. Such issues could not be solved by our theory, but would have to be addressed by the Company A employees who could extend the handbook modifying the proposed best practices to take into account industry-specific issues.

Relevance was assessed for both the inbound governance section and for the individual industry best practices from the proposed theory, as we evaluated how relevant the theory was to Company A in terms of addressed the company's needs of inbound governance. Evaluating the section as a whole, we found that some senior software developers already had informal solutions for open source component approval and reuse. However, as presented in the initial situation assessment, such informal practices were local and differed based on a developer's experience and understanding of open source governance. The pilot project team considered that using the part of our theory for inbound governance. This was considered to be highly relevant to the top management of Company A, which aimed at establishing an efficient and consistent inbound open source governance processes instead of the current local workarounds. Evaluating the best practices from the section, we observed that one of the most relevant best practices to the pilot project was *OSGOV-INBGOV-*

COMAPP-1. Define the component approval process, which would guide all software developers in the company in their open source component use eliminating the risks of non-compliance (caused by the current workarounds and lack of governance awareness). Another highly relevant best practice was considered to be *OSGOV-INBGOV-COMREU-5. Establish component reuse process*, which would enable the reuse of the previously used open source components.

Significance was assessed for both the inbound governance section and for the individual industry best practices from the proposed theory, as we evaluated the level of impact our theory had on Company A. Similar to the getting started section, evaluating the section as a whole, we could not fully assess how significant the impact of our proposed theory would be to Company A after the full roll-out across the whole company. Unlike the getting started section that was already used by the pilot project, which enabled the evaluation of the theory significance, the inbound governance practices were still being implemented at Company A, which did not allow us to evaluate the significance based on the actual use of the section as a whole across the company. Though we could not evaluate the significance of the complete section, we recognized that for both the pilot project team, the software developers and the legal counsel, one of the most significant aspects of inbound governance focused on the component repository (its establishment, maintenance, and use) that would enable an efficient open source component reuse at Company A.

Usefulness was assessed for both the inbound governance section and for the individual industry best practices from the proposed theory, as we evaluated how much value it added to Company A in solving the issues of inbound governance, as well as whether it enhanced the employee knowledge on these issues and their solutions. Similar to the evaluation criteria of significance, we could not assess the usefulness of our theory to the whole company during Case Study A, as the handbook was still being implemented. However, we found that the main issue to the usefulness of the inbound governance section was its lack of detail for some aspects of inbound governance, such as component integration, which was one of the subsections out of scope in our study, but that would be highly useful to Company A, according to the pilot project team. Another issue to the usefulness

of the inbound governance section was the difficulty of setting up tools with the support of the IT department (responsible for tool installations). For example, it would take several months for the IT department to approve and install open source governance tools, which forced the pilot project team to find workarounds for an early evaluation of the handbook best practices. A potential solution to this issue could be involving the IT department more in open source governance at Company A, giving them clear responsibilities. As to the evaluation of the individual best practices, we found that some best practices were not usefulness to Company A, such as the best practice *OSGOV-INBGOV-COMREU-4. Designate a role of responsibility for the component repository, in multiple places in the company* we had discussed earlier.

This concluded the evaluation of the inbound governance part of our theory at Company A, as well as our report on Case Study A. To presented the theory evaluation of the supply chain management governance at Company B in Section 4.5.2.

4.5 Case Study B

Case Study Profile – Company B

Summary:	Large German company operating internationally in enterprise software industry, and extensively using open source software in its products
Duration:	1 year (June 2018 - May 2019)
Location:	Germany (Hessen)
Maturity:	Basic open source governance in place, but no governance of suppliers
Evaluation:	Industry best practices for open source governance in Supply Chain Management

From June 2018 - May 2019, we extensively studied the open source use and governance at Company B. We focused our study on the centralized team, which we conditionally call Division B.1 though it is not a company division in the same sense as at Company A or Company B. This team worked company-wide on the issues of open source compliance, which was a sign of FLOSS governance maturity at Company B in comparison with Company A and Company B (as was planned during case study sampling). We conducted ten one- to two-hour interviews with managers, developers, Procurement, and compliance officers (in different locations in the German federal state of Hessen) using the questionnaire attached in Section C.2 in Appendix C.

We aimed at deeply evaluating the core part of our theory focused on open source governance in supply chain management (including preventive and corrective governance, as well as managing bills of materials). Given Company B's initial governance maturity, we were able to evaluate this more advanced part of open source governance. We aimed at guiding the implementation of our open source governance handbook at Company B, which would be followed by the theory evaluation of the selected handbook section, as well as select best practices from this section. We present the initial assessment of the open source governance situation at Company B in Section 4.5.1, followed by the

evaluation results of the proposed industry best practices for supply chain management governance in Section 4.5.2.

4.5.1 Initial Situation Assessment

Confirming our sampling criteria for Company B, we found that the company had basic open source governance in place, especially focused on inbound and outbound governance. Company B was the most mature in terms of open source governance among our three case study companies. The company had some formal governance concentrated at a central business unit called *Technical Compliance Department*. This department was tightly connected to the R&D department, legal counsel, procurement office, as well as production teams. At the time of the study, it was going through a rebranding as part of an overall reorganization of the company after a change in the top management. The department had existed for about 20 years dealing with various aspects of technical compliance related to software (not only open source software), dealing with open source and proprietary (commercial) software licenses, compliance process and automation, as well as other aspects of inbound and outbound governance. Technical Compliance Department created and maintained several centralized processes, such as the open source license clearance process for the components used by Company B's developers.

Reporting the initial situation assessment at Company C, we presented the details of the strengths, weaknesses, opportunities, and threats of using open source in Company B's products in the following subsection.

Strengths, Weaknesses, and Opportunities of Open Source Use in Products

Similar to the initial situation assessment of open source governance at Company A, we conducted ten interviews with Company B employees (for interview details see Table 4.1 – in its central organizational unit dealing with open source governance and compliance. As a result of these situation assessment interviews, we identified a number of strengths, weaknesses, opportunities, and threats

of open source governance at Company C, presented as follows:

- Strengths
  - Company B had a well-established, institutionalized, centralized, and company-wide department dealing with the basics of corporate open source governance and compliance – Technical Compliance Department
  - Company B had consistent processes in place for open source software usage including
    * inbound compliance review
    * outbound product review
  - Company B recognized open source governance and compliance as important issues to deal with company-wide
  - Company B was aware that the current open source governance process was covering only the basics, which left out the assessment of open source use resulting from the supplied code (including open source software that would come into the company from software purchased from third-party suppliers)
  - Technical Compliance Department did not assume that open source software use was correctly reported, tracked, or audited
  - Technical Compliance Department automated knowledge management and other aspects of open source governance using company-internal tools (wikis, license scanning tools, component management tools, etc.)
  - Technical Compliance Department provided answers to frequently asked questions around open source compliance to the stakeholder employees across the company
  - Technical Compliance Department provided limited education on the risks of ungoverned open source software use, as well as the existing guidelines Company B had for such use
  - A single point of contact was provided for outbound governance and external contacts (with suppliers and customers) for open source compliance questions.

- Weaknesses
  - There were unaddressed issues of open source governance and compliance
  - Technical Compliance Department's process for open source compliance (as part of inbound governance) were not well enforced, which led to possible workarounds by the developers under time pressure or unaware of the inbound governance process
  - Knowledge penetration of existing processes was not comprehensive
  - Most code from third-party suppliers was not reviewed
  - Inbound governance did not apply on the third-party supplied code as Company B only relied on contractual safeguards for potential open source compliance issues
  - Responsibilities in the governance process were not always clear, especially in the production teams, procurement department, and IT department
  - Company B made minimal contributions to open source software, and was not engaged in open source community leadership activities
  - Some tooling was lacking and inefficient, especially when dealing with the supplied code
  - There was no required or recommended format for bills of materials provided by suppliers (suppliers often submitted no BOMs or simple PDF documents with lists of open source components), which created compatibility issues hindered the efficient management of BOMs at Company B
  - No tooling or machine readable format was used for bill of materials (including open source components) of the supplied software.
- Opportunities
  - Technical Compliance Department and top management recognized the need for a more comprehensive open source governance process, especially focused on supply chain management

- Technical Compliance Department recognized the limited open source governance and compliance without employing more advanced tooling and was planning to look into tools for supply chain management
- Company B was getting invited to industry working groups dealing with open source governance, compliance, and security
- Technical Compliance Department had resources for student helpers who could help with license scanning and compliance review
- Introduction of the open source governance handbook could help improve and create new structured processes, rules and guidelines
- Company B was a large company with high negotiation power in comparison to most of its software suppliers, therefore they could enforce efficient governance among suppliers with little pushback
- Company B had the possibility to contribute to open source software communities to access the benefits of engagement
- Developing a process for the review of the supplied code could mitigate the risk of non-compliance with open source licenses (previously unidentified but being part of Company B's products)
- Open source components from the supplied code could be tracked and reused within the company without an additional compliance clearance process (after being cleared once), which would decrease in-house development costs, ensuring software consistency, and compatibility.

• Threats

- Supplied code which was accepted as a "black box" with only contractual safeguards against open source license non-compliance and other risks of the ungoverned open source software use

- Even when the supplied code came with a bill of materials detailing the used open source components, Company B did not verify the correctness of the reported BOMs, which left room for incomplete and incorrect BOMs
- Gray areas of responsibility for initiating a compliance review (in the case of supplier code) created a risk of open source software non-compliance
- Customers were becoming more aware of open source non-compliance risks asking for assurances of corporate open source governance
- Limited resources from management to the Technical Compliance Department could limit the efficient open source governance, especially for the supplied software
- Opportunity costs of the ungoverned open source use could be underestimated leading to the potential neglect of open source governance needs
- Lack of an evaluation process for the quality of the open source components coming through suppliers could put Company B at risk of exposure to vulnerabilities (e.g. security vulnerabilities)
- Unclear policy about open source software contributions in personal time created a risk of intellectual property loss
- Slow and burdensome governance process increased the likelihood that people would bypass it or refuse to adopt it
- No defined value-effort estimation was formalized, possibly leading to sub-optimal decisions about open source software use.

Initial Situation at Division B.1 – Technical Compliance Department

Division B.1 corresponded to Company B's Technical Compliance Department, which coordinated and oversaw the company-wide open source governance and compliance processes. Technical Compliance Department had one team tasked specifically with open source governance and compliance. This team included employees in the following roles: technical top manager, compliance manager, compliance officers, license reviewers, procurement officer / liaison.

We interviewed some of these employees to assess the initial situation of open source governance at Division B.1, and by extension at Company B. In addition, we also interviewed employees from outside of this core team to see the perspective of program and product managers, as well as software developers who were using open source software in their day to day work.

Analyzing our data (interviews, internal documentation, etc.) from Company B, we confirmed that the company had some formalized governance in place, especially focusing on the basic aspects like license compliance and inbound governance, but clearly lacked governance of the software supply chains. In this section, we summarized the highlights of the existing open source governance at Company B (mainly at Division B.1).

Technical Compliance Department had developed some company-wide rules and guidelines for open source use and governance, which ensured the consistency and knowledge sharing in all parts of the company (both in Germany and in other countries where Company B operated). However, these rules fell short of becoming an overarching open source governance policy that would cover all aspects of corporate open source governance. One of the reasons for this was the lacking structure for the existing guidelines, which is a prerequisite of a governance policy. The guidelines also excluded key aspects of governance, such as IP-at-risk analysis, component integration, or supply chain management. Discussing the guidelines for open source use and governance, the interviewed compliance manager recognized their lack of an official overarching open source policy. The guidelines were integrated into Company B's knowledge management system (set of wiki pages), which was available to all the employees at the company. Developers, managers, lawyers and other employees referred to these rules in their day to day product development, while the Technical Compliance Department maintained and updated these rules based on the new requests and precedent decisions.

Some of the key processes of open source governance in place before we introduced our handbook included:

- *Open Source Component Approval and Compliance Process*
- *Open Source License Interpretation Process*

- *Open Source License-Use Case Pair Documentation Process*
- *Compliance Automation Process*
- *Component Reuse Process*
- *Hiring Process with Focus on Open Source Competencies.*

Open Source Component Approval and Compliance Process provided a step by step process for developers to follow when requesting the approval of an open source component use in Company B products. The process is highly focused on open source license compliance. Developers used tools integrated into the product development environment to file open source component requests. The responsible team at Technical Compliance Department reviewed such requests referring to the guidelines and documented precedent decisions. The team would use compliance and license checking tools to approve or reject the use of the requested component. If approved, the developer would then refer to company guidelines for the required steps to fulfill license compliance for the to be used component. This workflow also checked other metadata of an open source component, including but not limited to the copyright information and use case. The process was being continuously optimized over the last 20 years, which shortened the component approval and compliance process from several months in the part to two weeks at the time of our assessment (though peak times of product delivery could cause minor delays). Reflecting industry best practices from our theory, the compliance review was not left to the end of the product development cycle (right before product release), but was integrated into the development process.

Open Source License Interpretation Process focused on the legal and technical interpretations of the commonly used open source licenses. Company B lawyers worked together with the Technical Compliance Department to interpret the consequences of using open source software under certain open source licenses. Such interpretation included both legal requirements that needed to be fulfilled in case of using the code under a given license, as well as the optimal technical requirements following from the latter. While the key licenses had been interpreted over time, new (to the company) licenses would be reviewed and interpreted when required by a developer or product man-

ager. License interpretation would first answer if the license could be used at all at the company, as some licenses would not be allowed in certain use cases (e.g. AGPL-licensed code in most products). The interpretation would then give a checklist of the mandatory technical requirements to fulfill for the compliant open source use. Updates were added to the company-internal wikis on the topic, which were regularly (in weekly meetings of the Technical Compliance Department, R&D department, product and project teams) shared with product development teams and other stakeholder employees.

Open Source License-Use Case Pair Documentation Process enabled Company B to create and maintain matrices of open source licenses and the use cases, for which these licenses are approved. License interpretation and license-use case pair documentation helped Company B to maintain lists of white-listed and black-listed license for given use cases (e.g. AGPL use not allowed in most products, but allowed for certain internally used tools). Such pairs are confirmed in the license review board (an inter-organizational team hosted at Technical Compliance Department), documented in centralized wikis and shared across the company. This matrix is currently documented in an Excel sheet, however Company B aimed at further automation and at direct integration in the development process.

Compliance Automation Process guided Company B efforts towards open source governance and compliance automation. This process helped Technical Compliance Department capture and prioritize automation needs at the company, collect and assess tooling options that would meet different requirements, evaluate several tools, plan the required budget and write the business case for purchasing the selected tools. Currently, Technical Compliance Department was assessing tools and automation options for the basic compliance review of the supplied code from third-party suppliers. The automation efforts at Company B started about 10 years ago and scaled up ever since.

Component Reuse Process provided software developers across the company with a database of the approved, previously used, and tracked open source components in all the products of Company B. The process would guide a developer in checking if an open source component he wanted to use

has already been used somewhere else at the company, which would mean it had gone through the open source component approval and compliance process. One of the essential checks the developers needed to perform was checking the version of the open source component to make sure they reused the exact component that was already cleared in the past. Component reuse process was also linked to the Excel sheet resulting from the open source license-use case pair documentation process. Company B also maintained a master data management system, where a developer could look up the specific products where a given open source component was already used.

Hiring Process with Focus on Open Source ensured that the Technical Compliance Department could get involved in the open source governance related competence evaluation of the potential software developer hires. For example, this process would enable one of the compliance officers to participate in the hiring process of a software developer. The compliance officer would evaluate the open source governance and compliance awareness of the applicant, which would affect the overall applicant evaluation.

Beyond the above-mentioned formalized processes, open source governance was also somewhat integrated into the software procurement process. Technical Compliance Department would help the procurement teams in selecting software vendors with open source compliance in mind. Company B developed several templates for software vendors evaluation in terms of their use of open source and its governance. Technical Compliance Department would also work with the procurement officers / liaison and lawyers to define open source related clauses in supplier contracts. Such clauses would define the responsibilities of the suppliers in case of any detected open source license non-compliance. These clauses would serve as limited safeguards against potential non-compliance caused by the supplied code used in Company B products. Such safeguards were limited as shifting risks of non-compliance to the suppliers could not be realistic in case of smaller suppliers. However, this contract review process was not a regular process, nor did it include any audits of the actual code supplied to Company B. Most contracts also did not require suppliers to provide bills of materials of the open source components for the supplied software. Some suppliers would provide BOMs in

Word or PDF formats, but their completeness or correctness was not checked. At the same time, few customers of Company B, in turn, asked for open source component BOMs for the sold software products. However, recently such requests by the customers were becoming more frequent, which was one of the motivations for Company B to collaborate with us and implement our governance handbook section on supply chain management. We presented the evaluation of the latter in the following Section 4.5.2.

4.5.2 Evaluation of Supply Chain Management

We conducted the guided implementation of the supply chain management section of our handbook at the pilot project at Division B.1 (Technical Compliance Department) of Company B. This section captured the industry best practices we had identified by analyzing the expert interviews at companies with an advanced understanding of corporate open source governance. The topic of supply chain management in the context of FLOSS governance was the focal subtopics of our proposed theory. To test this part of our theory we guided the implementation of the respective handbook section at a production level project (pilot project at Division B.1) at a company that already had basic governance in place, but lacked in its the more advanced aspects as was supplier management.

Similar to Case Study A, after assessing the initial situation of open source governance at Company B and after having developed the supply chain management part of our theory, we organized a workshop at Company B with our primary contact employees and their colleagues from the pilot project at the Technical Compliance Department chosen for the theory evaluation due to its established role at the company as the central hub for all open source governance and compliance issues. Another reason for the handbook evaluation at the Technical Compliance Department was their identified need for formal open source governance in managing the suppliers and the supplied software. At the time of the handbook introduction at Company B, one of the compliance managers was looking for potential tools that could be used for checking open source license compliance of the supplied code. At the same time, Division B.1 was working on a company acquisition, whose

software assets would also need to be checked for compliance, which would need a process similar to the FLOSS governance one for suppliers. As was the case with Case Study A, during this workshop we presented the developed handbook section to the stakeholder employees, going into the details of select best practices and workflows that interconnected several practices. We called such workflows process templates, as a company using the handbook would need to adjust and modify the proposed process workflows or create new ones that would fit the company-specific processes and guidelines. For example, Company B considered extending the supplier management processes to also cover company acquisition. It's important to note that the latter was addressed in a different pattern by our theory, which did not match the handbook implementation pattern by Company B. We used such pattern matching in theory evaluation.

After we provided the handbook section on supply chain management to the Technical Compliance Department, together with our partners from Company B, we then selected the pilot project and the team that would carry out the implementation of the handbook. After the introductory workshop, we met with the pilot project team that consisted of the technical top manager, a compliance manager, a compliance officer, and a procurement officer. During the guided implementation the technical top manager oversaw the work of the pilot project team and provided strategic input. The compliance manager and the compliance officer dealt with the operational side of the handbook section implementation, as well as its integration into the existing processes at Company B. The procurement officer worked directly with the existing and future suppliers following the handbook best practices on ensuring the open source governance in the software supply chains. In a follow-up workshop with the compliance manager, the compliance officer, and the procurement officer, we identified the specific best practices for implementation from the supply chain management handbook section. The pilot project team prioritized the proposed industry best practices based on their needs, as well as on their estimated applicability in the scope of our case study.

In the preparatory stage of the handbook implementation, the pilot project asked us clarification question about select best practices. Among other questions, we discussed the best practice B.2.5

(*OSGOV-SUCHMA-SCMPRO-5. Use tools to automate supplier management*) and the further details and guidance on choosing the right tooling. We also discussed the best practice B.3.1 (*OSGOV-SUCHMA-PREGOV-1. Choose the right supplier*) and the pilot project team's need to deviate from it, while applying it not only to the suppliers, but also applying it to the Company B's recently acquired company. After clarifying the questions of the pilot project team, we observed the specific best practices that the pilot project team had selected for implementation. Some of these practices included:

- *OSGOV-SUCHMA-PREGOV-1. Choose the right supplier* (see B.3.1 in Appendix B)
- *OSGOV-SUCHMA-PREGOV-1.1. Assess open source governance and compliance awareness and maturity* (see B.3.2 in Appendix B)
- *OSGOV-SUCHMA-PREGOV-1.2. Request supplier certification or self-certification* (see B.3.3 in Appendix B)
- *OSGOV-SUCHMA-CORGOV-1. Audit your supply chain* (see B.4.1 in Appendix B)
- *OSGOV-SUCHMA-BOMMAN-1. Identify open source components and metadata from the supply chain* (see B.4.7 in Appendix B)
- *OSGOV-SUCHMA-BOMMAN-2. Track, document and update BOM in a consistent and complete manner* (see B.4.8 in Appendix B)
- *OSGOV-SUCHMA-BOMMAN-3. Have a backup of open source components hosted by yourself* (see B.4.9 in Appendix B)
- *OSGOV-SUCHMA-BOMMAN-4. Use machine readable and standard format for BOM upon software supply* (see B.4.10 in Appendix B).

The pilot project started with the implementation of some of the above-mentioned best practices right away, while some others were postponed. For example, the best practices B.3.2 (*OSGOV-SUCHMA-PREGOV-1.1. Assess open source governance and compliance awareness and maturity*) and B.3.3 (*OSGOV-SUCHMA-PREGOV-1.2. Request supplier certification or self-certification*) were

applied first. The pilot project team followed these best practices to create a supplier questionnaire, which they sent to several of their suppliers to assess their current level of open source governance and compliance awareness and maturity. See two excerpts (the first and the last pages) from the artifact with this supplier questionnaire in Figure 4.9 (the first page) and in Figure 4.8 (the last page). See the full artifact in Section F.3 in Appendix F.

FOSS Compliance for Licensors

COMPANY is certified according to standard ISO norms. Therefore it requires its licensors to be compliant with the same norms or has to conduct regular supplier audits with its licensors so check whether the minimum standard processes are in place to ensure standard compliance, quality and security of the products.

1. Licensor Information
2. Quality Management
3. Level of Open Source Awareness
4. Technologies used in the Development Process
5. Integration of Third Party Software into Product
6. Support of Sales Process
7. Requirements to use Licensor's Products using Free and Open Source Software
    a) Licensor shall provide Bill of Material in one of these suitable formats:
    b) Licensor shall provide Source Code of Open Source Components:

1. Licensor Information

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

Figure 4.7: Case Study B – Excerpt from Supplier Questionnaire on FLOSS Governance and Compliance Awareness and Maturity – First Page

7. Requirements to use Licensor's Products using Free and Open Source Software

a) Licensor shall provide Bill of Material in one of these suitable formats:

	Format	Specification
Minimum Requirement	PDF document	specs to be requested by COMPANY
Standard Requirement	Machine-readable `debian/copyright` file	https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
Best/ recommended Requirement	COMPANY specific SPDX Format	

b) Licensor shall provide Source Code of Open Source Components:

Minimum Requirement	where this is required by Open Source License (e.g. GPL family of licenses)
Standard Requirement	all components where Source Code is available
Descriptions of the Location: (FTP, GIT, ...)	

Figure 4.8: Case Study B – An Excerpt from the supplier questionnaire on open source governance and compliance awareness and maturity – Last Page

Looking at the last page of the supplier questionnaire on open source governance and compliance awareness and maturity in Figure 4.8, the pilot project team outlined three levels of maturity a supplier could attain:

- minimum level

- standard level
- best level.

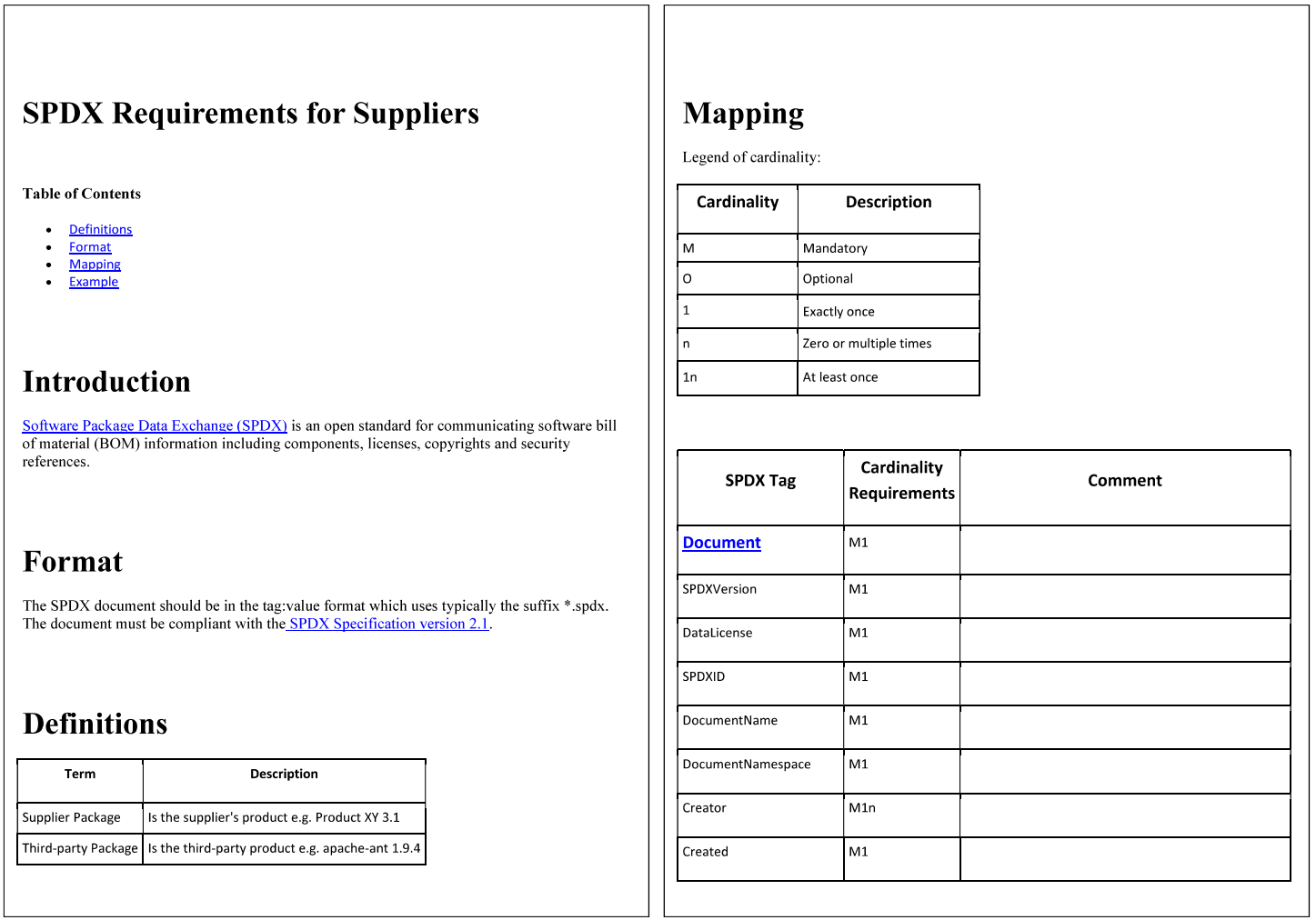
Accordingly, the pilot project team defined three formats of providing the information on what open source components had been used in the supplied code. The three formats corresponded to the three different requirements to the suppliers (called licensors within Company B):

- PDF document as the minimum requirement
- Machine readable *debian/copyright* file as the standard requirement
- Company-specific SPDX document as the best / recommended requirement.

Though our theory proposed assessing the suppliers' governance and compliance awareness before they are selected (see the best practice B.3.1 *OSGOV-SUCHMA-PREGOV-1. Choose the right supplier*), the pilot project team at Company B deviated from the proposed pattern and used the developer supplier questionnaire to assess the governance and compliance awareness of the existing suppliers first (planning to take action if the suppliers lacked governance and compliance awareness).

To look at another implementation pattern at Company B, the best practices B.4.7 (*OSGOV-SUCHMA-BOMMAN-1. Identify open source components and metadata from the supply chain*) and B.4.10 (*OSGOV-SUCHMA-BOMMAN-4. Use machine readable and standard format for BOM upon software supply*) were implemented early on. To identify the open source components and their metadata supplied to Company B, the pilot project team designed a list of open source component reporting requirements for the suppliers using the machine readable SPDX[12] format. See an excerpt from the artifact capturing this in Figure 4.9. See the full artifact in Section F.4 in Appendix F.

[12] Software Package Data Exchange (SPDX) – `https://spdx.org/`



SPDX Requirements for Suppliers

Table of Contents

- Definitions
- Format
- Mapping
- Example

Introduction

Software Package Data Exchange (SPDX) is an open standard for communicating software bill of material (BOM) information including components, licenses, copyrights and security references.

Format

The SPDX document should be in the tag:value format which uses typically the suffix *.spdx. The document must be compliant with the SPDX Specification version 2.1.

Definitions

Term	Description
Supplier Package	Is the supplier's product e.g. Product XY 3.1
Third-party Package	Is the third-party product e.g. apache-ant 1.9.4

Mapping

Legend of cardinality:

Cardinality	Description
M	Mandatory
O	Optional
1	Exactly once
n	Zero or multiple times
1n	At least once

SPDX Tag	Cardinality Requirements	Comment
Document	M1	
SPDXVersion	M1	
DataLicense	M1	
SPDXID	M1	
DocumentName	M1	
DocumentNamespace	M1	
Creator	M1n	
Created	M1	

Figure 4.9: Case Study B – Excerpt from SPDX Requirements Specification for Suppliers

At the time of the evaluation at Case Study B, we received early feedback from the suppliers to the sent questionnaires. The large suppliers were familiar with open source governance, thus they did not find it difficult to fill in the governance awareness questionnaires. They also were familiar with the SPDX format, though most did not use it for the supplied code to Company B. They started considering switching to the SPDX format for bills of materials of the supplied software. On the contrary, the small suppliers were puzzled by the open source governance questionnaire, in some cases escalating the Company B request to their CEOs. After the clarification by the pilot project, the smaller suppliers understood the request, but opted for meeting the minimum requirement of submitting a PDF document with the bill of materials of the used open source components. At the time of the evaluation, the pilot project was still waiting for more responses from the suppliers.

There were some best practices from the supply chain management section of the handbook that the pilot project decided to implement later. For example, the best practice B.4.9 (*OSGOV-SUCHMA-BOMMAN-3. Have a backup of open source components hosted by yourself*) would be implemented only after the suppliers provided their complete bills of materials including open source components and their metadata.

Another artifact the pilot project team created during the handbook implementation outlined a proposed compliance process was for Company B following our theory's best practices B.4.12 (*OSGOV-SUCHMA-LICCOM-1. Review identified open source components and metadata for license compliance*), B.4.13 (*OSGOV-SUCHMA-LICCOM-2. Review license obligations in the context of supply chain management*), and other practices from the handbook, including *OSGOV-INBGOV-COMAPP-1. Define the component approval process*, presented in Table 3.9, and *OSGOV-OUTGOV-LICCOM-1. Ensure license compliance*, presented in Table 3.18.

This resulted in an artifact that illustrated a proposed compliance process, developed by a compliance officer at Company B following the above-mentioned best practices from our theory, and extending them aspiring to achieve continuous compliance. Our theory did not find this to be an industry best practice, as many of the interviewed experts during theory building deemed it to be

unrealistic given the currently available compliance tools. Though we did find that companies want to have tools to meet a requirement of continuous compliance, as presented in our paper on industry requirements for FLOSS governance tools [68].

Figure 4.10 presented the final version of the proposed continuous compliance process at Company B. The incremental versions that led to this final version are presented in Section F.5 in Appendix F.

The legend for the figure included:

- *iData* – a Master Data Management system, which was used to manage Company B's product catalogue. The catalogue contained technical dependencies between different products and their third-party products (TPP – third-party components including open source software).
- *PCI Scanner*, which identified requested (known) TPPs and gave as output the scanning results to be used to manage the BOM stored in iData repository.
- *TP Vault* – a Repository that contained requested TPPs (sources and binaries).
- *TPP Fetcher* – an internally developed tool that collected TPP metadata (component names, versions, licenses, copyrights, etc.) from different sources within the built environment. Sources could be dependency managers/declarations or source code scans. It fetched TPP files (source code and binaries) that belonged to a TPP via package managers. It uploaded TPP metadata and TPP files to the TPP Interface.
- *TPP Interface* – an internally developed tool that took TPP metadata to create requests and uploaded TPP files to TP Vault, which triggered the TPP review process.

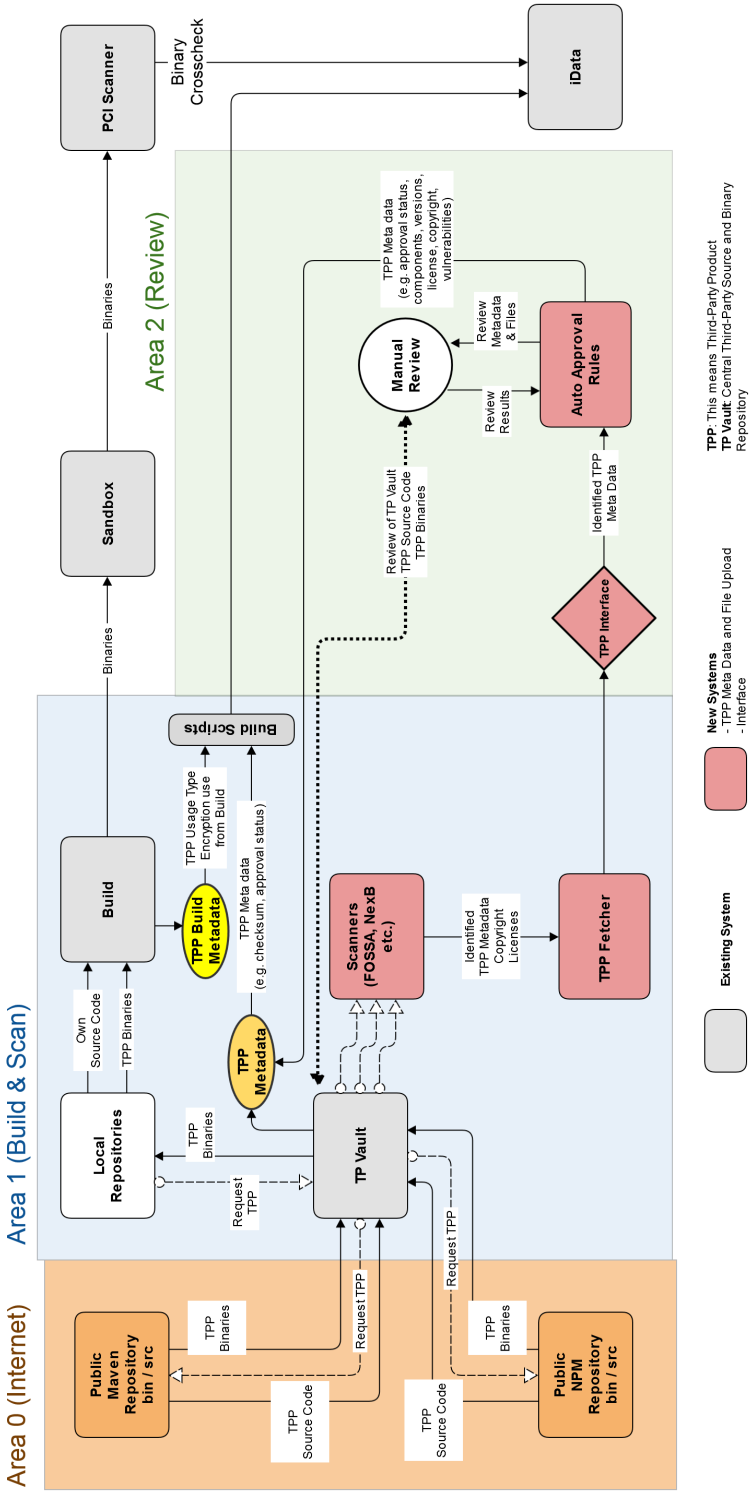


Figure 4.10: Case Study B – Proposed Continuous Compliance Process at Company B

After observing the guided implementation of the supply chain management section of the governance handbook based on our theory, we went on to evaluate the transferability of the proposed theory's part on *supply chain management*. Similar to Case Study A, we used the same evaluation criteria defined in the evaluation case study protocol, which we presented in Appendix E, including Completeness, Variability, Structure, etc.

Completeness was assessed for the supply chain management section as a whole, as it evaluated whether the section had an adequate beginning, middle, and end, as well as whether it lacked any practices the company needed when applying the handbook. The employees tasked with the implementation of the supply chain management part of our theory reported in our follow-up interviews that the handbook section on the topic did not have any gaps and matched their expectations in terms of covering all the key aspects of the supply chain management in the context of FLOSS governance. One exception was the security aspects of supply chain management. The pilot project team expected to see industry best practices for security-related aspects of open source use. However, as this was out of the scope of our study, we provided pointers for other materials that cover the issue, while taking note that this issue to the section completeness also came up during theory evaluation at Company A.

Variability was also assessed for the supply chain management section as a whole (similar to completeness), as it evaluated whether the section had a balanced mixture of concepts for the topic and not overly focused on a single concept. During Case Study B, we observed that the balanced design of this part of the proposed theory translated into an equal coverage of different supply chain management concepts, such as SCM policy, SCM process, preventive governance, corrective governance, and BOM management. The variability of the section enabled Company B to prioritize the best practices from the subsections. The implementation pattern illustrated that Company B did not focus on the subsection on SCM policy at first, as they were first trying the specific best practices on preventive governance, corrective governance, and BOM management. The pilot project team decided to revisit the topic of SCM policy in the later stages of handbook implementation, which

deviated from the proposed pattern from our theory suggesting to start with the policy. The pilot project team noted that no concept was singled out and presented in more detail than others. Such an assessment led us to the positive evaluation of the variability criteria of this part of the tested theory.

Structure was assessed for both the supply chain management section and for the individual industry best practices from the proposed theory, as we evaluated how well-structured both the section and the individual practices were. For the section as a whole, we evaluated whether its different parts were structured in a logical and interconnected manner. Similar to the evaluation at Company A, the pilot project employees who were implementing the handbook at Company B appreciated the interconnecting links between individual best practices within the supply chain management section, as such links created natural workflows that could be made into company processes and were already ingrained into the theory, therefore, making it easier to apply at the company. However, a major structure-related issue with the links the compliance manager from the pilot manager noticed was about the lacking mechanism of going back to a best practice after one had followed a link. It was hard to find the original best practice after clicking away from it. We recognized this issue, and while it was not directly caused by our theory, but rather its presentation in the PDF format, we did consider this as a structural issue that negatively impacted our theory evaluation. This navigation issue was not special to the supply chain management section. It impacted the whole handbook, but we could not find a solution to the issue using the PDF format. Instead, we recommended using a specialized tool for the governance handbook, such as the one by Editive we had presented in Section 4.4.2 on the getting started evaluation. As to the structure of the individual best practices, Company B employees involved in handbook implementation saw the value of using the structured presentation format for the industry best practices from our theory – the Context-Problem-Solution pattern format that made the practices more digestible. This was a similar assessment to that observed in Case Study A.

Comprehension was assessed for both the supply chain management section and for the individ-

ual industry best practices from the proposed theory, as we evaluated how well the theory answered the problems companies with only basic governance would have, as well as whether the proposed best practices went into enough detail on their respective issues. Evaluating the section as a whole, we found that some best practices were too complex for the pilot project team at Company B. The main issue was that in order to ensure governance for software supply chains, the company would have to follow the handbook section in full, but the pilot project team wanted to implement select best practices only at first. To address this, the pilot project team wanted to know which best practices are the most essential ones, and which ones are only optional. As our theory did not make such distinction between best practices, we could not provide a list of best practices whose implementation would ensure the minimal, yet comprehensive solution to open source governance related issues for Company B's software supply chains. Instead, the pilot project went on to choosing the combination of best practices that they considered to be addressing all their early governance needs in a comprehensive manner. Unlike the evaluation at Case Study A, Company B employees did not see an issue in the handbook section being too general, abstract, or not customized enough for Company B. They recognized that, as we had intended, they would need to adjust and customize the handbook to their needs in order to build a comprehensive supply chain management process at their company. This was the case for the best practices that were implemented at Company B, such as the best practices B.4.7 (*OSGOV-SUCHMA-BOMMAN-1. Identify open source components and metadata from the supply chain*) and B.4.10 (*OSGOV-SUCHMA-BOMMAN-4. Use machine readable and standard format for BOM upon software supply*) that were customized for Company B. Their implementation resulted in the Company B specification with SPDX requirements for suppliers that needed to report their use of open source components and their metadata supplied to Company B.

Understandability was assessed for both the supply chain management section and for the individual industry best practices from the proposed theory, as we evaluated how understandable the theory and its representation in the handbook format were to the employees implementing and us-

ing the handbook. We focused on assessing the understandability of both the intentions and the specifics of the proposed theory. Evaluating the section as a whole, we found that the pilot project employees did not have any issues with reading and understanding the handbook, neither in terms of the language nor in terms of the content. This observation was different from that at Case Study A, where some employees had understandability issues. This could be explained to the existing basic open source governance at Company B with a heavy focus on compliance. Having some governance expertise helped the pilot project team in understanding the content and the intention of the proposed industry best practices for the supply chain management. However, one issue was raised by the compliance officer, who suggested adding a glossary of the best practices and their key concepts at the beginning of the section. We recognized that such a glossary could improve the understandability of the section, especially to the occasional user of the handbook. (e.g. developers, lawyers). No understandability issued were voiced about individual best practices.

Applicability was assessed for both the supply chain management section and for the individual industry best practices from the proposed theory, as we evaluated how well our theory could be applied to a company with a different context from that at the expert companies involved in theory building. We evaluated how generalizable the supply chain management part of the theory was, as well as how much the evaluated best practices needed to be adjusted to become applicable at Company B. Evaluating the section as a whole, we found that the biggest challenge for the applicability was the limited number of proposed workflows / process templates that combined the best practices in the section. The pilot project team used the two proposed process templates, but had to design more processes to guide different users of the handbook through supply chain management, as was intended by our theory. We did not aim at covering all possible uses and workflows of the best practices in a given section, leaving this to the customization and extension by the companies using the handbook. One of such workflows developed by the pilot project went over the proposed best practices and integrated the governance process into the company-internal development process. This resulted in a proposed workflow of continuous compliance, whose final version

was presented in Figure 4.10. This artifact was created by building on top of the best practices B.4.12 (*OSGOV-SUCHMA-LICCOM-1. Review identified open source components and metadata for license compliance*), B.4.13 (*OSGOV-SUCHMA-LICCOM-2. Review license obligations in the context of supply chain management*), and other practices from the handbook. Similar to the evaluation at Company A, some best practices were out of the scope for Company B at this early stage of supply chain management. An example of such a practice was the best practice B.4.9 (*OSGOV-SUCHMA-BOMMAN-3. Have a backup of open source components hosted by yourself*), which was postponed to the later stages of ensuring supply chain management governance at the company.

Relevance was assessed for both the supply chain management section and for the individual industry best practices from the proposed theory, as we evaluated how relevant the theory was to Company B (and, by proxy, to its employees) in terms of addressed the company's needs of supply chain management governance. Evaluating the section as a whole, we found that the employees in the pilot project and in the Technical Compliance Department as a whole recognized the relevance of supply chain management in the context of open source governance. As a matter of fact, the compliance officer from the pilot project team had already considered several issues of supply chain management, such as using bills of materials for open source governance and related tooling, before our guided implementation of the handbook at Company B. The best practices in the subsection of BOM management was, therefore, of special relevance to the pilot project team. Similar to our evaluation at Case Study A, Company B also planned to leverage our handbook and research project with our university as a way of demonstrating the high relevance of the governance topic in the industry to their higher management, hoping for more resources, which the department lacked as presented in the initial situation assessment.

Significance was assessed for both the supply chain management section and for the individual industry best practices from the proposed theory, as we evaluated the level of impact our theory had on Company B. Evaluating the section as a whole, we could not fully assess how significant the full impact of our proposed theory would be to Company B (after the handbook section was

implemented as a whole). Evaluating the implementation of the select best practices (small scale implementation), we recognized that the handbook significantly strengthened Company B's efforts towards FLOSS governance of their software supply chains. The highest impact was achieved through the implementation of the subsection on BOM management as part SCM. The compliance manager and the compliance officer reported that during the whole 2017 they were looking at ways of addressing the lack of the SCM governance at Company B. However, they needed to develop a governance framework following industry best practices, which was possible only to a limited extent given Company B's resources. The pilot project team highlighted the high significance of our handbook (and its SCM section) for Company B, as a reference to an academic theory of industry best practices. As a result of using our handbook, Company B did not need to find the industry best practices anew, but rather focused on developing their own SCM framework based on our theory.

Usefulness was assessed for both the supply chain management section and for the individual industry best practices from the proposed theory, as we evaluated how much value it added to Company B in solving the key issues of SCM governance, as well as whether it enhanced the employee knowledge on these issues and their solutions. Similar to the evaluation criteria of significance, we could not assess the usefulness of our theory as a whole due to the implementation of the select practices at Company A in the early stage (during our evaluation). However, we did evaluate how useful the chosen best practices were to the company during Case Study B. When choosing the best practices with the highest priority for early implementation, the pilot project team implicitly evaluated the perceived usefulness of these practices. The subsections on the preventive governance and on the BOM management were considered to be the most useful to Company B. The pilot project team, therefore, implemented multiple best practices from these subsections, including the BP B.3.2 (*OSGOV-SUCHMA-PREGOV-1.1. Assess open source governance and compliance awareness and maturity*) from the preventive governance subsection, and the BP B.4.7 (*OSGOV-SUCHMA-BOMMAN-1. Identify open source components and metadata from the supply chain*) from the BOM management subsection. At the early stage of handbook implementation, Company B saw

more value in preventing future issues with supply chain management by following the preventive governance best practices, while sorting out the already supplied code using the BOM management best practices. In the later stages, the pilot project team planned to implement the rest of the SCM subsections, including that on the corrective governance, as well as on the SCM policy and process, whose usefulness we could not evaluate during Case Study B.

This concluded the evaluation of the supply chain management part of our theory at Company B, which was the focal topic of our theory.

4.6 Case Study C

Case Study Profile – Company C

Summary:	Large German company operating internationally in the automotive industry, and using open source software in its products
Duration:	9 months (April 2018 - December 2018)
Location:	Germany (Bavaria), France, USA
Maturity:	Basic open source governance in place, but limited inbound and outbound governance, and no governance of suppliers
Evaluation:	Industry best practices for General Governance, Inbound Open Source Governance, and Outbound Governance (failed evaluation)

From April 2018 - December 2018, we extensively studied the open source use and governance at Company C. Our first and major focus was Division C.1 (Interior). Using the Division C.1 as a benchmark, we went on to assess open source use and governance situation in other areas, namely Division C.2 (Safety). With the help of a colleague, we conducted eight one- to two-hour interviews with managers, lawyers, software developers, and other stakeholders (in different locations in the German federal state of Bavaria, in France, and in the USA) using the questionnaire attached in Section C.2 in Appendix C.

We aimed at evaluating three parts of our proposed theory at Company C covering the parts of our theory untested in Case Study A or Case Study B – industry best practices for open source governance focused on general governance, inbound open source governance (focused on component reuse), and outbound governance. We aimed at guiding the implementation of our open source governance handbook at Company C, which would be followed by the theory evaluation of the selected handbook sections, as well as select best practices from these sections. However, unlike Case Study A and B, we encountered a problem at Company C, which led to a failed evaluation

of our theory. One of the reasons was the mismatch in expectations of what a corporate open source governance handbook should look like and be used at the company. Another reason was company politics and the pushback from the company internal team working on the issues of open source governance. We discussed the details of the failed evaluation in Section 4.6.2. As a result, our project ended in December 2018, and we were not able to fully evaluate how our theory was used at Company C.

4.6.1 Initial Situation Assessment

Confirming our sampling criteria for Company C, we found that the company and its divisions had little open source governance in place (more mature governance than at Company A, but less mature than at Company B). Some informal governance existed at parts of the company, but none centralized or company-wide. The only two formalized groups working locally on open source governance and compliance were:

- *Open Source Compliance Team (OSCT)* – responsible only for the license compliance issues of open source software
- *Processes and Tools Team* – responsible for the inbound review of open source components and tools, as well as in charge of the OneCompanyC process that involved both software quality and legal topics related to FLOSS governance.

In addition to the above-mentioned local teams, there was one cross-organizational group working towards company-wide open source policy – *Global Experts Team*. We observed division-specific political tensions in this group, which was partially responsible for the failed evaluation of our open source governance handbook. We discuss the latter in Section 4.6.2.

Reporting the initial situation assessment at Company C, we presented the details of the strengths, weaknesses, opportunities, and threats of using open source in Company C's products. We then presented our initial situation at Division C.1 and that at Division C.2.

Strengths, Weaknesses, and Opportunities of Open Source Use in Products

Similar to the initial situation assessment of open source governance at Company A and Company B, we conducted eight interviews at Company C (for interview details see Table 4.1 – in its divisions C.1 and C.2. As a result of these situation assessment interviews, we identified a number of strengths, weaknesses, opportunities, and threats of open source governance at Company C, presented as follows:

- Strengths
  - Company C had a consistent process in place for open source software usage (inbound review)
  - Company C had a consistent process for compliance checking in place (outbound review)
  - Company C did not make the assumption that open source software use would always be correctly reported and employed a code scan
  - The Open Source Compliance Team and Legal Department made an effort to educate people on the risks of ungoverned open source software use
  - The Open Source Compliance Team offered training on their processes
  - A single point of contact was provided for external open source software compliance questions
  - Review processes were designed to be business-unit-agnostic.
- Weaknesses
  - There was no clear mandate from top management to prioritize open source software governance, and therefore the policy was largely determined at the division or business unit level
  - Knowledge penetration of existing processes was extremely limited

  - Knowledge penetration of the risks associated with ungoverned open source software use was unknown and probably limited
  - Knowledge retention of processes was minimal among parts of the organization which were not regularly interacting with the Open Source Compliance Team
  - Most code was probably not reviewed
  - Compliance was largely treated as a reactive, one-time event, rather than a proactive, iterative activity
  - Institutional knowledge was not shared between groups
  - Responsibilities in the governance process were not always clear
  - Company C made minimal contributions to open source software and was not engaged in leadership activities
  - The open sourcing process was too difficult, leading to lost opportunities
  - The inbound review process was too complex for non-distributed use cases
  - There was a lack of guidance on contributing to open source software.

- Opportunities

  - Cross-BU open source software group showed awareness of threats; implementation of investigated solutions would address several issues
  - Introduction of open source software governance and compliance handbook could help establish structured processes, rules, and guidelines where they did not exist
  - Company C developed software which had the potential to become a de-facto standard among OEMs, given timely open sourcing
  - Company C had the possibility to contribute to open source software communities to access the benefits of engagement
  - Developing a process for participating in bug- or feature-bounties could allow the development of important components at a reduced cost
  - Greater participation in appropriate consortia would give Company C a more leading role in the direction of development

- Participation in open source software development could lead to Company C's expertise being recognized within the industry
- Participation in open source software development could help attract and retain quality developers
- Contributing to open source software communities could improve development efficiency and quality of used open source software components (for generic / non-differentiating components)
- Open source components could be reused within the company decreasing in-house development costs, ensuring software consistency and compatibility
- Open source software could be a platform of interaction and collaboration between product teams within and across Company C divisions, resulting in knowledge and resource sharing.

• Threats

- Supplier code which was accepted "as is" created a risk of open source software non-compliance
- Gray areas of responsibility for initiating a compliance review (in the case of supplier code or mergers and acquisitions) created a risk of open source software non-compliance
- Decisions made at the product team level creates the risk of open source software non-compliance and loss of intellectual property when teams did not understand the risks associated with non-compliance
- Decisions made at the product team level led to inconsistent decisions, reducing opportunities for reuse
- The lack of a process to maintain data on open source software component use, to compare this list against new vulnerability information, and to investigate the impact of vulnerabilities on existing products put Company C at risk of exposure to vulnerabilities

- The lack of a process to evaluate the quality of open source software components put Company C at risk of exposure to vulnerabilities
- Unclear policy about open source software contributions in personal time created a risk to Company C intellectual property
- Slow and burdensome governance process increased the likelihood that people would bypass it or refuse to adopt it
- No defined value-effort estimation was formalized, possibly leading to sub-optimal decisions about open source software use
- The lack of a centralized repository with open source software components and license data limited component reuse
- Open source software capabilities were concentrated around a few developers, which created the risk of the process being interrupted if one of these developers was unavailable for an extended period of time.

Initial Situation at Division C.1

We found that Division C.1 (Interior) had the most developed governance process within Company C. The Open Source Compliance Team (OSCT) was formed within a business unit in Division C.1, before transitioning in 2014 to a service organization at the divisional level. The governance process was first formalized in 2009 and has since been adapted to respond to changing requirements. In addition to this, there were initiatives in place to improve FLOSS governance, for instance, the company process OneCompanyC (process title changed to hide company's identity) which involved both quality and legal topics and touched on FLOSS governance topics as well.

Open Source Compliance Team at Division C.1

The Open Source Compliance Team was a service organization located within Division C.1 but with a mandate to serve all division and business units. It occupied a position between the engineering and legal departments, and was tasked with:

- Inbound review
- Outbound review.

OSCT tried to make its presence known across Company C, for instance, at an internal software development conference. However, knowledge of the team was not widespread throughout the company. Some business units have adopted training materials provided by OSCT to assist in compliance but this was not uniform across Company C. Nonetheless, there had been increased demand for their services as FLOSS usage was spreading in new countries (where Company C started operating), technology areas, and business units. The questions the team would receive started addressing more situations than before, and there were more questions.

Inbound Review at Division C.1

The inbound review concerned all code which was brought in to Company C from external sources. In practice, the inbound review was primarily applied to open source software components which were to be incorporated into Company C products. Changes in the use case or updating to a new version of the software should trigger a new inbound review.

The process was initiated with a review request by the team wishing to use the external code. It was submitted as a ticket to the Open Source Compliance Team, which then identified the most restrictive applicable license and delivered a report of the implications of the use of the software and advice for addressing the concerns.

By design, the final decision on how to respond to the concerns was devolved to the team which had submitted the request, except in cases where the license was rejected outright for legal reasons. An example of this might be the GPLv3 license as a linked component in a commercial product, as opposed to a GPLv3-licensed software used as a tool (e.g., GCC).

The strengths of the process included:

- A consistent process was available for all inbound open source software usage
- Training was available on the process

- Delegation of decision allowed the business needs to be considered.

However, there were also weaknesses with the process:

- Architects did not have knowledge of inbound reviews conducted for other teams, which:
  - Limited opportunities for reuse
  - Prevented early avoidance of components which were known to be problematic
  - Resulted in an increased effort for the Open Source Compliance Team
- There was no assurance that the decision process would be consistent, or conducted in any systematic way
- Use of the process was voluntary, which meant that in all probability not all open source software was evaluated
- Open source software components were not scanned for quality (e.g., with Clockwork)
- There was no centralized database with records of component use, which not only hindered reuse but also limited the response to vulnerabilities discovered in components after the release of the product
- Lack of a process to re-evaluate already used components limits vulnerability response
- Open source components in supplier code were not documented in a standard format, and compliance largely depended on the supplier.

Outbound Review at Division C.1

The outbound review concerned all of Company C products which would be released. Internal projects and demonstrations (demos by R&D) were not necessarily reviewed. Ideally, the process should be applied at intervals, but in practice, it was applied right before release, though before the code had been frozen for release.

The process was initiated by a request for review initiated by the product team. The request was made as a ticket to the Open Source Compliance Team. OSCT used a change control board to analyze the ticket for data completeness. If the data was complete, it was assigned to a reviewer. If it was not complete, OSCT requested additional information.

The reviewer performed a license compliance review on all known open source software components and scanned the product using a compliance tool to identify any undeclared copied code. The product team was supplied with a report listing the risks and proposing solutions. The decision was, as in the case of the inbound review, left to the product team.

The strengths of this process were identical to those of the inbound review, while the disadvantages of the process were:

- No institutional knowledge of previous scans was used: knowledge of previous scans rested with the Open Source Compliance Team, and the response of product teams was not captured in a reusable way. This was problematic because:
  - Code scans often identify canonical solutions which can look like copying (false positives) and these patterns will be reported each time, resulting in an increased effort for development teams to resolve known problems
  - Each version of a product must be fully re-analyzed, instead of undergoing a delta review, which is time-consuming. Because of the effort involved:
    * Not all releases would be reviewed
    * As software updates became more rapid and common, the process would be applied to a decreasing proportion of releases
    * The final product would not be reviewed immediately before release, but while development was still ongoing
- There was no assurance that the decision process would be consistent, or conducted in any systematic way

- Use of the process was voluntary, which meant that in all probability not all open source software was evaluated
- There was no link between the review process and the creation of the bill of materials, leading to duplication of effort
- Product teams were dissatisfied with the process:
  - The process was seen as time-consuming and complicated
  - Product teams would have liked issues to be prioritized according to risk to enable business decisions
- The quality of material submitted for outbound review was often poor, either missing important information or containing irrelevant data, which delayed the work of the Open Source Compliance Team.

Open Source Software Usage – Component Use in Products at Division C.1

Open source software was being used in products. Product teams initiated both inbound and outbound reviews from the Open Source Compliance Team. Compliance was intended to be an ongoing, iterative process, but as described in Section 4.6.1 (detailing the outbound review process), time constraints rarely allowed teams to go through this process as expected.

The strengths of the process included:

- There was a clear understanding, in principle, that compliance was more effectively obtained by design rather than as an afterthought.

The weaknesses included:

- Compliance was not treated as an iterative process but is often postponed until near release:
  - Compliance was treated as a low importance activity due to high pressure

  - Teams were often reactive, driven by customer requirements or 'ticking a box' rather than viewing FLOSS compliance as part of the development lifecycle

- Outbound review was often not conducted directly before release, but often at 60-80% completion
- Developers found reports difficult to interpret, due to the lack of prioritization of risks, lack of institutional knowledge of previous decisions, and insufficient business knowledge to interpret the risks in the context
- Management lacked an overview of priorities and risks.

Open Source Software Usage – Component Use in Platform at Division C.1

Open source software was being used in platform development (different from product development). Platform teams initiated inbound review but did not make use of outbound review because the platform was considered an incomplete product. Not all parts of the platform would necessarily be used in a particular product, and linking might change. Therefore product teams had responsibility for outbound reviews, which might include parts of the platform in addition to their own work.

The advantage of this approach included:

- Product-focused outbound reviews allow the use case and context to be understood by the team requesting the review, and the decisions to be made by the team with the business understanding of the situation.

The concerns which stem from this process included:

- Scanning the code only at the product phase meant that it was possible for non-compliant code in the platform to escape notice until near product release, when addressing the issue would create a bottleneck

- Product teams found it difficult to acquire an overview of all the FLOSS included in the platform, resulting in:
  - Difficulties creating the final bill of materials
  - Inefficiencies in reuse, as product architects could not default to using FLOSS components which were already used in the platform.

Open Source Software Usage – Component Use in R&D, Demos at Division C.1

Open source software was widely used in R&D and demos. Both open source tools and components were used. In this context, the inbound review was rarely conducted and the outbound review was not conducted. This was based on the belief that the current governance process was designed primarily for product development and was unsuited to the style of development used in R&D and demonstrations. The particular concerns with the formal process were:

- The PDF reports were not considered useful because they were lengthy due to every issue being listed (including relatively unimportant ones such as missing license headers), without prioritization
- The reports took too long to be delivered
- Developers would prefer to automate the process and identify and resolve simple issues (such as missing headers) themselves
- The process was unsuited to the ongoing and rapid development found in R&D and demos, because it operated on a single, fixed (release) version of a product.

Open source software usage decisions were made by developers, who had received some training on different licenses. The policy was for crucial technology and joint technology to be kept separate from the main code, and to use only basic functionality from open source software components. A matrix of use cases and licenses had been requested to aid in this decision, but had not yet been provided. An open source software component list was maintained (open source components, versions,

licenses) and this information was provided with the software when the project was handed over to another part of the company.

The strengths of the process included:

- Developers appeared to have a good awareness of the need for open source governance, even in the use case of internal use
- Developers had some understanding of licenses and particular risks associated with linking
- This use case had a lower risk than products in the case of non-compliance.

The weaknesses of the process included:

- Developers were making decisions without a systematic process or the guidance of use cases
- Developers might not be completely aware of the effects of tool licenses on outputs
- No architectural model displaying open source software components was supplied to other parts of the business, which might hinder the transfer of knowledge
- No standard format was used to convey information about open source software components which were used.

Open Source Software Usage – Tool Use at Division C.1

Company C made use of a number of FLOSS tools such as the GCC compiler, the Linux kernel, and Eclipse IDE. Depending on which part of the organization was involved, tool review could be done as an inbound review by the Open Source Compliance Team or by the Global Experts Team (a cross-division internal team of experts working with open source). However, it seemed that neither team did significant tool review, leading to the conclusion that FLOSS tool use might be largely ungoverned. Furthermore, although there were localized attempts to encourage reuse. For instance, in Division C.1 the Processes and Tools Team created mirrors of FLOSS repositories. However, there was no systematic approach which would be readily searchable across Company C.

Initial Situation at Division C.2

Going beyond Division C.1 (Interior), we studied the initial situation of open source governance at Division C.2 (Safety), which focused on safety-related products of Company C that included FLOSS components. Open source software was being used in Division C.2, in the creation of demo products and in R&D. Prototypes were provided to other teams, who rewrote the software following ISO specifications, thus open source compliance had been dealt with in a relatively simple manner. Currently, there is no outbound review, although this might change as the size of projects and the need for speed might mean that it would become impractical to completely rewrite software (when going from R&D to production). The latter was also observed in our initial situation assessment at Division A.1 of Company A.

Use of open source software was preferred over building or buying, largely because the complicated functionality contained in components could rarely be improved upon by independent efforts. Mature projects were preferred. However, in situations where the only open source solution was under a restrictive license (with a copyleft effect), a supplier might be hired to write an alternative. The handling of suppliers was left to external support working on software solutions, so it was unknown what open source governance processes were in place for incoming supplier software.

There was no formal training for new developers on FLOSS usage or governance, although there was an annual presentation about licenses and obligations delivered by the legal department. New developers were generally given projects which did not require the use of new (previously unused at Company C) libraries.

When a new library was selected, it was sent to the Processes and Tools Team for an inbound review. The team checked the library against an internal traffic light system (manually) – a system with certain white-listed and black-listed licenses. If the library was approved, it was compiled and made available on the server. A matrix was maintained with the division projects and the open source components used by them. This matrix was also linked to the licenses of the used FLOSS components. The licenses were rechecked when new versions of the software were acquired. This

system had the following benefits:

- It was easy to use and could be used by employees with an understanding of the development environment
- Only libraries which were marked ”green” were used, reducing the risk caused by ”yellow” libraries in the wrong context
- Developers couldn’t add libraries on their own so the vetting process had to be used for inbound FLOSS components
- A matrix of projects and used libraries was systematically maintained and updated
- Component reuse was encouraged by the use of a centralized collection (a division-wide database with limited properties)
- There was good communication between developers and thus informal knowledge sharing about FLOSS components in use.

However, there were also some disadvantages to the system:

- Header files were not systematically checked, only the license text, which meant that open source components with a mix of licenses (which is a very common case) might not be correctly identified
- No code quality checks for open source components was performed
- Evaluation of governance and compliance tools was handled by the IT team
- No outbound review was applied.

Open source software contributions were one area of governance which was of particular concern. Contributions were not being made by Company C employees, except for bug reports (that was also rare). Even in cases where the code had been fixed internally, the fix would not be given back to the community. This appeared to be due to a lack of clear direction about how developers could seek approval to contribute. Some disadvantages of the situation were:

- When fixes were only applied internally, patches had to be reapplied with new releases of the same open source component
- There was an increased risk of development proceeding in a direction which made the software less useful to Company C
- Company C was perceived as isolated, rather than as part of the community.

The first disadvantage was also perceived as such by some developers at Company A (especially in Division A.2).

Division C.2 also did limited corporate open sourcing (when a company shares its in-house developed software as open source). They developed some components which would be useful to their customers, and which would aid in cooperation with their partners, if open sourced. However, open sourcing had been stalling for two years despite management approval (though not fully governed overall). This was because the open sourcing process was seen as complex and unclear to developers, especially when there had been no previous engagement with the Open Source Compliance Team. The specific problems included:

- The Open Source Compliance Team was only found through recommendations by word of mouth among developers
- Interactions with the Open Source Compliance Team were viewed as unproductive, with each question resulting in additional questions which required a level of licensing expertise that developers lacked
- The legal department couldn't answer questions because they couldn't book hours for this project.

The first two points resulted from the lacking open source governance at Division C.2 in contrast to that in Division C.1, which hosted the Open Source Compliance Team and was more mature in terms of FLOSS governance overall.

In addition to the concerns with contributions and open sourcing which were described, we also identified some other potential gaps in corporate FLOSS governance:

- Knowledge about FLOSS components in use was not formalized or documented in a public way (within the company), and depended on personal relationships
- It was unknown if inbound supplier components were scrutinized for FLOSS compliance (we didn't find indications that it was)
- Quality of FLOSS components was not examined
- Vulnerabilities of FLOSS components were not tracked
- The unclear policy about open source software contributions in personal time created the risk of developers sharing Company C's intellectual property (caused by the lack of governance processes)
- In-house code developed was not scanned for compliance
- It was assumed that developers were aware of open source compliance (though no training was provided).

4.6.2 Failed Evaluation

Similar to Case Study A and Case Study B, after assessing the initial situation of open source governance at Company C, we were planning to guide the implementation and then to evaluate parts of our theory in the format of the governance handbook sections. Company C was chosen to be more mature (in terms of FLOSS governance) than Company A, but less mature than Company B, which would enable us to evaluate the parts of our theory on general governance, inbound governance (focused on component reuse), and outbound governance. In parallel to the initial situation assessment at Company C, which was being conducted with the support of a research colleague, we developed and extended these parts of our theory, presenting them in the handbook format used in the other two case study companies. We then delivered the handbook (with the relevant sections) to

our primary contacts at Company C – a project manager and a manager of processes and tools, who we planned. Following our statement of work, we expected that our primary contacts would introduce the handbook at a pilot project within Company C, and connect us to the pilot project team for the evaluation.

Following the situation assessment, we considered Division C.1 (Interior) to be the place to find a potential pilot project for the handbook implementation and evaluation. Division C.1 has some of the most advanced FLOSS governance within Company C. The Open Source Compliance Team had created a process by which both inbound and outbound reviews could be performed, though only locally within the division. There were several positive points about the process, most notably that decisions took business needs into consideration, code scans were performed to identify hidden compliance violations, and the entire process was meticulously documented through a ticketing system. There were, however, several key weaknesses with the current approach, including:

- open source component reuse and knowledge sharing
- lack of universal governance processes and process avoidance
- limitations of existing local governance across the company.

One concern related to the lack of shared knowledge and component reuse within the organization. Because of the lack of a consistent format (such as SPDX) for conveying open source component usage, product teams struggled to understand OSS use in the platform and OSS use in supplier components, and to produce bills of materials. Component reuse was hindered by the lack of a common database of decisions and components. Open Source Compliance Team reports were difficult to interpret because they contained repetitive, minor issues which could be excluded if the resolutions were communicated, and because project teams lacked the knowledge to prioritize issues. To date, the emphasis had been largely on inbound review for license compliance, with some outbound license compliance review, and awareness of other governance issues appears to be limited to a few people.

Another concern focused on the ways in which the governance processes were bypassed or avoided due to confusion or because it was perceived as burdensome. Moreover, the processes which had been described in the initial situation assessment related largely to Division C.1, and were not widely employed across the company. Even within Division C.1, it did not appear that tool review was routinely done, and products were not being reviewed in the release state. It was also unknown to what extent the process was followed for code which entered Company C through mergers and acquisitions. The existing processed were largely designed for products with set releases, and were not suitable for the more rapid development cycles used by the R&D department for demonstrations, for example. While the outbound review process was described as difficult and time-consuming by some employees, the process for open sourcing was described as all but impossible, requiring years of effort.

The third concern addressed the gaps in the existing open source governance. Inbound suppliers were trusted to declare OSS components, without the assurance of either a code scan or a certification (e.g. OpenChain self-certification). OSS components were not systematically scanned for vulnerabilities by all teams, and there did not appear to be a procedure in place to re-evaluated products after release in response to any newly discovered OSS component vulnerabilities. Furthermore, the decision-making process regarding OSS component use did not appear to be subjected to a formalized value-effort estimation.

During our handbook implementation, we planned to address the above-mentioned concerns, which would allow us to evaluate parts of the proposed theory.

Before delivering the handbook, we also provided some initial guidance and pointers for our primary contacts at Company C to consider in preparing for the handbook implementation. Company C was looking at purchasing tools for open source governance, which our primary contacts hoped could help introduce company-wide governance processes in an efficient manner. In support of this effort, we provided an overview of tools for FLOSS governance and compliance that Company C could consider in preparation for the handbook implementation. In line with our theory's best

practices A.3.4 (*OSGOV-GETSTA-PROANA-1.3. Select and use governance tools for automation*) and *OSGOV-INBGOV-COMREU-12. Use tools to create, update and maintain component repository (for tools focused on component reuse)*, we provided the tooling overview developed by a research colleague, which is presented in Figure 4.11.

Fossology

Fossology is one of the popular tools for OSS compliance and focuses on detection of licenses, copyrights, or export controls information. The basic workflow using Fossology is dividable in the following steps:

License scanning

To detect licenses, copyrights, or export controls Fossology using different pattern recognition methods.

License review/clearing

One of the key features of FOSSology is the user interface to review license findings in order to determine the exact licensing of a file. A review of the findings is necessary because the unequivocal detection of a license is not trivial. For instance, the license text can be modified or a complete unknown license is detected. The review/clearing results are stored in a database and can be reused for other scans.

Report generation

In the last step, a report with all detected components with their licenses, copyright, and export restriction information can be created. This list is called bill-of-materials (BOM). The usual representation of a BOM is in the form of the machine-readable SPDX standard.

SW360

SW360 on the other hand, helping users by establishing a central hub for software components in an organization. SW360 allows for

- tracking components used by a project/product,
- assessing security vulnerabilities,
- maintaining license obligations,
- enforcing policies, and
- generating legal documents.
- Integration with other tools and data sources (e.g. license scanner, static code analysis, build infrastructure, etc.)

SW360 doesn't provide necessary functionalities for license clearing by itself; instead, it can trigger a clearing process in FOSSology and import the resulting clearing reporting.

The is another example where different compliance tools can be used together, or depend on each other, e.g., OSS Review Toolkit (ORT) and ScanCode.

ScanCode from NexB

ScanCode is another popular exemplar of a license scanner like Fossology, with similar functionalities. No further explanation here because it's very similar to Fossology.

OSS Review Toolkit

The goal of the OSS Review Toolkit (ORT) is to verify Free and Open Source Software license compliance by checking project source code and dependencies. ORT analyzing the source code for dependencies, downloading the source code of the dependencies, scanning all source code for license information, and summarizing the results. It uses data from a projects build system (Maven, Gradle, etc.) to determen all components and it's dependencies and for the actual license check it triggers one of four supported license scanner. ScanCode is here the recommended option. ORT also using SPDX as a format to exchange results with other tools.

Figure 4.11: Case Study C – Tooling for FLOSS Governance and Compliance

In addition to the tooling overview, we also provided a list of the common industry requirements (not specific to Company C) for FLOSS governance and compliance tools based on our previous research [68], which our primary contacts at Company C could use in choosing the right tooling for the company-wide governance and compliance. See an excerpt from this list of requirements in Figure 4.12.

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.

Figure 4.12: Case Study C – Tool Requirements for License Compliance of FLOSS Components for Company C

However, after the handbook delivery, handbook implementation and evaluation issues arised. Our primary contacts from Company C were not satisfied with the handbook and did not want to

proceed with its implementation. To discuss the issues, we organized a workshop with the case study colleagues to go over the handbook and their concerns. Analyzing this meeting we identified the following key issues that might have potentially caused the failed implementation and evaluation at Company C:

- expectations from the handbook were not met, as the open source governance handbook was not customized for Company C
- internal politics at Company C created tensions between two groups working internally on open source governance.

The first issue was in the mismatch of the expectations of what a corporate open source governance handbook should look like. Our statement of work with Company C clearly outlined that we would develop a theory on industry best practices based on the data gathered from industry experts, which abstracted from individual companies and resulted in a general handbook of open source governance. However, our industry partners at Company C voiced a different expectation after the handbook was delivered. They expected a customized handbook that could be readily applied at Company C, which was not the case as we provided different versions of the abstract handbook to all three case study companies. Each company would then customize the handbook during the implementation phase fitting it to their needs and processes. Unwilling to customize the handbook on their own, our partners at Company C limited our theory evaluation in Case Study C.

The other challenge to the handbook implementation and theory evaluation was caused by company politics and tensions between Division C.1 and Division C.2, each of which had teams working on open source governance that could apply to the whole company. We had observed these tensions and related conflicts of interest during the initial situation assessment, but did not expect them to affect our work that would aid the company as a whole in establishing company-wide FLOSS governance. However, after our handbook delivery and before we could start the guided implementation, the company-internal issues around governance efforts blocked our project from accessing a pilot project at Company C for theory evaluation.

To conclude, with the limited access to the company-internal information, we could not claim that our observations were the only reasons for the failed evaluation, but rather the ones that we were aware of.

5

Conclusion

This chapter concludes the dissertation with a discussion of the implications of our findings to both the academic community and to practitioners. We discuss the future research questions can be explored by peer researchers on open source governance, and on the use of open source in companies more broadly. We also address the limitations of this study, both during theory building and during theory evaluation. We discuss the issues related to the internal validity and to the external validity of our work.

5.1 Discussion

Our research goal was to build and evaluate a theory of industry best practices for corporate open source governance. In the course of this study, we started by assessing the state of the art in the literature, which helped us understand the landscape of the domain and its key subtopics. We then conducted a qualitative survey of the industry expert knowledge and derived industry best practices for FLOSS governance. The identified governance framework and industry best practices constituted the proposed theory. We presented our theory in the actionable format of a pattern handbook. Finally, we evaluated our theory through a multiple-case case study at two companies, where we guided the implementation of different parts of our handbook and evaluated the respective parts of the proposed theory. The evaluation resulted in identifying some issues to the completeness, variability, structure, comprehension, understandability, applicability, relevance, significance, and usefulness of select best practices from our theory.

Assessing the state of the art literature, we reviewed 87 papers on the topic of governance, which identified insights and practices on the following core issues of corporate open source governance:

- Getting Started
- Inbound Governance
- Outbound Governance
- General Governance.

As a large subtopic within Inbound Governance, we considered Supply Chain Management as a separate topic of FLOSS governance.

Taking the state of the art review as our basis, we developed a theory of industry best practices for corporate open source governance, which partially confirmed the findings from the literature, while partially rejecting some findings and extending the current understanding of FLOSS governance in companies. We contributed to both the academia and industry by applying scientific methods, such as qualitative survey and qualitative data analysis, to address our core research question on

how companies should govern the use of open source software in their products. A novel topic, this specific subject was little researched as its own domain. Previously, researchers focused mostly on the topics of either open source community governance, open source license compliance as part of corporate open source governance. In our theory, we built upon the research on open source license compliance, but also covered some rarely studied aspects of FLOSS governance. Our theory covered the following topics, subtopics, and industry best practices of corporate open source governance in detail:

- Getting Started with Open Source Governance
- Inbound Governance
  - Component Approval
  - Component Repository
- Supply Chain Management
- Outbound Governance
- General Governance.

Our theory provides industry best practices for companies with no governance in place that want to introduce open source governance after recognizing the risks of the ungoverned use of open source software in their products. We then suggest best practices for inbound governance dealing with how OSS components get into the company. We present in detail two key topics of inbound governance – component approval that focuses on the process software developers need to follow when adding open source components in their products, and component repository that focuses on the reuse of the previously used open source components. We then present the focal topic of our study – Supply Chain Management that focuses on dealing with software suppliers and open source software in the supplied code, as well as topics of the bill of materials management, and license compliance in the context of software supply chains. We further cover outbound governance dealing with the outward-facing issues of FLOSS governance, such as license compliance of the shipped

products, contribution management, etc. Finally, our theory presents industry best practices on the general and crosscut topics of open source governance, such as the establishment and operation of an open source program office.

Some of the proposed industry best practices in our theory confirm the findings from the literature. For example, the best practice *OSGOV-GETSTA-TRAPOL-1 Establish FLOSS governance policy for the transition period* confirms findings by Lerner & Tirole [93], while the best practice *OSGOV-SUCHMA-PREGOV-1.1. Assess open source governance and compliance awareness and maturity* confirm the recommendations by [82].

Other proposed industry best practices in our theory reject some findings from the literature or propose previously unknown ones. For example, while Copenhaver [31] suggested having an employee role specifically for ensuring license compliance – the Open Source Software Compliance Officer (OSSCO), our theory proposes delegating this task to the open source program office (OSPO) – a group of employees that are able to cover not only the legal aspects of open source compliance, but also its business and technical aspects. The best practice *OSGOV-GENGOV-GOVMAN-3. Establish an open source program office* presents the specifics of establishing an OSPO. One explanation for this difference between our approach and that of Copenhaver could be explained by Copenhaver's strong focus on the open source license compliance compared to our broader focus on corporate open source governance as a whole, where compliance is only one of many tasks the OSPO needs to manage.

After building the theory of industry best practices for corporate open source governance, we presented it in the actionable format of interconnected patterns that formed a handbook of FLOSS governance. We presented excerpts from two major parts of the handbook in the appendix of this dissertation – that on Getting Started in Appendix A and that on Supply Chain Management in Appendix B. We hope that such a practitioner-friendly presentation format will make our theory more applicable at companies, which will increase the impact and the relevance of our research contributions.

To evaluate the proposed theory, we guided the implementation of the Getting Started and Inbound Governance best practices from our theory at Case Study A, where we found that most practices were well-structured, comprehensive, and applicable. However, we found that some best practices, such as *OSGOV-GETSTA-TRAORG-4. Start small, then replicate - define the scope of the transition process* lacked understandability, while some others lacked usefulness in the context of Company A, such as *OSGOV-GETSTA-TRAORG-1. Establish a board of stakeholders to organize the transition*. Instead of the latter, Company A opted for another best practice – *OSGOV-GETSTA-TRAORG-2. Designate the transition manager*, extending which the transition manager would assume the responsibilities of the board of stakeholders in the early phase of the transition towards FLOSS governance. We then guided the implementation of the Supply Chain Management best practices from our theory at Case Study B and evaluated our theory using the same quality criteria as at Company A.

To conclude, future research on the topic of corporate open source governance can further extend the scope of the proposed theory, especially focusing on the topics of outbound governance and general governance. Other studies can be conducted to further evaluate the current theory, either using the case parts of the governance handbook (to replicate our findings) or using the currently unevaluated parts of the handbook. Furthermore, FLOSS governance can be studied directly in the context of software supply chains, without the involvement of an OEM company at the end of the supply chain, whose perspective we took in our study. FLOSS governance automation and product models are also topics of high relevance to the industry, but not extensively researched – a gap that future researchers can address. Finally, going beyond the corporate use of open source software, researchers can study why and how companies contribute to open source communities, as well as why and how companies create and lead open source projects.

5.2 Limitations

5.2.1 Limitations for Theory Building

Our study faces several limitations. For theory building, we follow Guba [61] in assessing the trustworthiness of our research through the following quality criteria:

- Credibility
- Dependability
- Confirmability
- Transferability.

Credibility is the degree to which we can establish confidence in the truth of our findings in the context of the inquiry. In our case, some potential sources for credibility limitations include the reliability on our research group's professional network for the sample of companies with an advanced understanding of open source governance, and the main mode of data collection being semi-structured interviews. To address these issues, we performed two rounds of peer debriefing, asked three colleagues to review the study design, and incorporated the feedback from our colleagues from within our research group. We also conducted data triangulation relying not only on expert interviews, but also on other primary materials (e.g. company guidelines for FLOSS governance).

Dependability is the degree of consistency of the findings and traceability from the data to the results. The usual limitations related to dependability include the lacking documentation of research protocols, research artifacts, and process documentation, as well as the limited traceability from data to the built theory. We addressed these issues by saving the raw interviews, their transcriptions, and metadata in the course of the study. We also documented our qualitative data analysis using QDA tools that enable fine-grained traceability from the primary materials to the resulting theory and its concepts.

Confirmability is the degree to which the authors are neutral towards the inquiry and their potential bias effect on the findings. Some of the limitations related to confirmability include the potential researcher bias, the bias introduced by the research funding, and the researcher assumptions formed prior to the study. We addressed these issues by following the research method constructs carefully, not trying to interpret the methods in a biased way. We also had a second coder analyze parts of our data, for example on the topic of getting started, which improved our original QDA coding based on input from the second coder [99].

Transferability is the degree to which the findings of our study hold validity in other contexts. Some of the limitations related to transferability include the over-reliance on examples from the data that could limit the generalizability of the theory, and the over-generalization based on the small sample. To address the issues of transferability we conducted a multiple-case case study to evaluate how generalizable our proposed industry best practices would be to companies with no governance in place and their different contexts.

5.2.2 Limitations for Theory Evaluation

Our theory evaluation also has its limitations both related to the method used and to the research process. Method related limitations include the typical issues associated with case study research, such as the limited of generalizability of the results outside of the studied cases, and the limited replicability in case of the poor research process documentation.

For theory evaluation, we follow Gibbert, Ruigrok & Wicki [56] in assessing the rigor of our case study research informed by Yin [157] through the following validity and reliability criteria:

- Construct validity
- Internal validity
- External validity
- Reliability.

Construct validity of a procedure refers to the quality of the conceptualization or operationalization of the relevant concepts emerging from the data. Potential limitations related to the construct validity of case study research include an ambiguously defined or implemented research process that could lead to poorly defined or measured concepts, limitations in data collection, as well as the researcher bias and subjectivity in case of the lacking research documentation. We addressed these issues by following a well-established and widely used case study method by Yin [157]. Before starting the case study, we created a detailed case study protocol that would guide us through the research process and reduce the researcher bias associated with sudden changes to research design in the course of the study.

Internal validity or logical validity [30] refers to the causal relationships between variables and results. Internal validity is essential for explanatory case studies, not for the descriptive ones [157]. For the explanatory part of the study (ours was a hybrid of a descriptive and explanatory design), potential limitation issues include whether the researcher provides a plausible causal argument, logical reasoning that is powerful and compelling enough to defend the research conclusions, and the issues related to data analysis instruments. We addressed these issues by following the data analysis construct of pattern matching proposed by Yin [157], which helped us evaluate the best practices from our theory with their implementation patterns at case study companies [38] [42]. Given our research goal of theory evaluation, by design, we focused on establishing logical reasoning behind for the assessment of the predefined quality criteria of the proposed theory.

External validity refers to the generalizability of the proposed theory in settings different from those in the research leading to the given theory [20] [110]. The key limitation related to the external validity of case study research is that case studies do not allow for statistical generalization. However, case studies are not devoid of generalization, instead, they provide analytical generalization, which refers to the generalization from empirical observations to theory, rather than a population [157]. In our study, we do not claim that our theory evaluation results can be generalized to a population of companies establishing open source governance, but we do suggest that in the context of the

studied companies, we observed and reporting how the proposed industry best practices could be implemented and used. To address this limitation, we also provided a clear rationale for the case study sampling and reported the case study context of Company A, B, and C in detail, which would allow the reader to understand our sampling choices [30].

Reliability refers to the absence of random error, enabling other researchers to arrive at the same insights when conducting a new study along the same steps [38]. The key limitation related to the reliability of case study research is the lacking transparency of the case study design and process, which can hinder future replication studies. To address this limitation, we provided a detailed case study protocol we had developed before conducting the case studies and using in the course of the study. We attached the case study protocol in Appendix E. We also archived the data we had collected in the course of our case study.

A

FLOSS Governance Handbook – Selected Practices for Getting Started

Appendix A presents an excerpt from the open source governance handbook's section on getting started with FLOSS governance. The handbook presents the findings of our research cast as a collection of interconnected patterns. This presentation format enables higher applicability of the research results. This appendix also includes process templates – example workflows connecting

various best practices on getting started, which further improves the practical applicability of our results. See Figures A.1, A.2, A.3, A.4 and A.5 for process templates.

A.1 Transition Organization

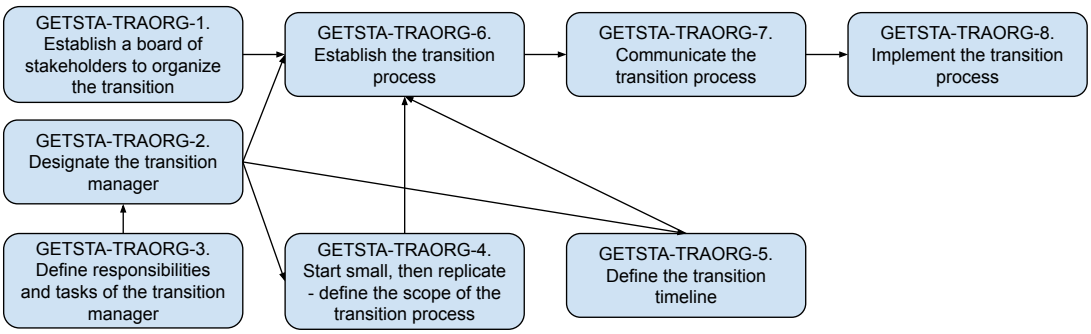


Figure A.1: Transition Organization Process Template 1

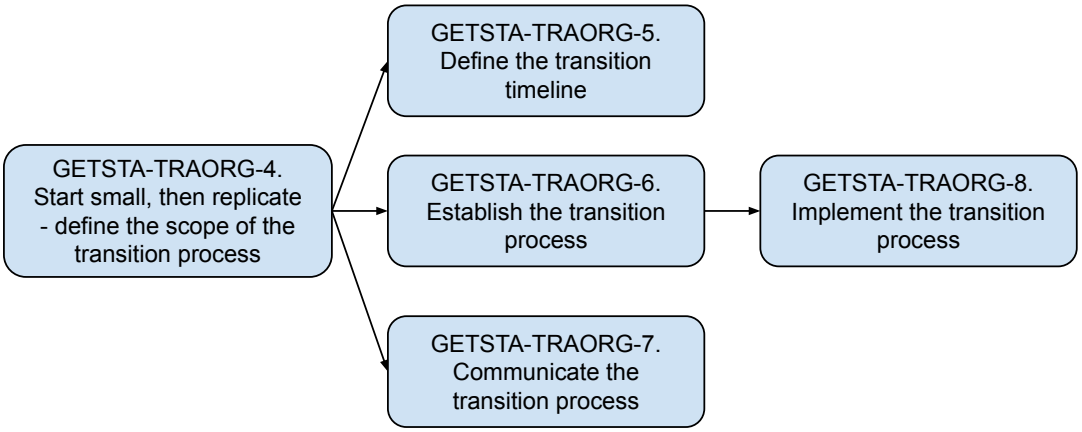


Figure A.2: Transition Organization Process Template 2

A.1.1 Establish a board of stakeholders to organize the transition

Name	Establish a board of stakeholders to organize the transition
Actor	Top management
Context	Your company came to recognize the importance of FLOSS governance. You decided to regulate your use of open source software in products using FLOSS governance best practices.
Problem	Before rolling out an overarching FLOSS governance process, you need to review all the existing products that include open source software components. Where and how do you start?
Solution	To start reviewing your existing products and their software components, you need to follow best practices on getting started with FLOSS governance. As a first step, establish a board of stakeholders to organize the transition from ungoverned FLOSS use to structured FLOSS governance. Your transition board should include the current users of open source in the company, decision makers regarding FLOSS use and those to be responsible for FLOSS governance in the future. For the transition board, consider the following employees: • senior developers (known internally for their open source use and competency) • engineering managers (usually de facto decision makers on FLOSS matters) • lawyer (responsible for FLOSS license clearance and related issues) • business/product managers • software architect • software procurement officer.

The transition board should be inclusive and transparent, open for any interested stakeholder to join. The board should not require full-time engagement of all the members. However, it's important to → *designate the transition manager* - a responsible role and person for the transition, and to → *define responsibilities and tasks of the transition manager*.

A.1.2 Designate the transition manager

Name	Designate the transition manager
Actor	Top management
Context	After → *establishing a board of stakeholders to organize the transition* towards regulated FLOSS governance at the company, you need to delegate the management of the transition process to a responsible person.
Problem	It can be hard to find the right person to oversee the transition, because of organization complexity, lack of FLOSS competency, limited human resources etc. However, it's essential to start the transition with a designated transition manager in charge to ensure smooth and timely transition. What's the best practice of choosing the transition manager?
Solution	The transition manager must already be involved with FLOSS use (and preferably decision making) in the company. It is preferable to choose a mid-management or senior development role to manage the transition, because of their understanding of the whole product or product line that uses open source components.

A junior developer might be more knowledgeable with open source, but not as competent in understanding and managing the overall transition process for the whole pilot project. A high level manager might lack clear and detailed understanding of open source use in products. During the transition, it's recommended to allocate 30-50% of the designated employee's time for transition management tasks. The transition manager coordinates transition board meetings and performs other → *defined responsibilities and tasks of the transition manager*.

A.1.3 Define responsibilities and tasks of the transition manager

---

Name	Define responsibilities and tasks of the transition manager
Actor	Top management
Context	In parallel to → *establishing a board of stakeholders to organize the transition* and → *designating the transition manager*, you must ensure the operation of the transition board and its manager.
Problem	As transition happens at a pilot project in the company, employees need clear guidance in transition related tasks and responsibilities to avoid wasting time and to ensure smooth and timely transition. How should the transition board and its manager operate?
Solution	Define clear responsibilities and tasks of the transition manager. The main task is to coordinate the transition process at the pilot project, while includes tasks and responsibilities to: ⇒ *define the scope of the transition process* ⇒ *define the transition timeline*

⇒ *establish the transition process*

⇒ *communicate the transition process*

⇒ *implement the transition process, while ensuring the efficient and timely implementation of the transition process*

⇒ *serve as central point in transition for any questions from the affected employees*

⇒ *manage the transition board in the process.*

A.1.4 Start small, then replicate - define the scope of the transition process

---

Name	Start small, then replicate - define the scope of the transition process
Actor	Transition manager
Context	Before → *establishing* and → *communicating the transition process*, the transition manager needs to define the scope of the transition process.
Problem	Depending on the company, the scope of the transition differs. Is there a common approach to defining a transition scope?
Solution	Chose a pilot project within the company with a product that uses open source software. For the pilot do not choose a project based on its complexity, but rather on its criticality and level of FLOSS use. If the project is critical and uses significant open source components (and preferably is B2C / customer facing), you have a good candidate. The transition manager identifies two candidates for pilot projects and chooses the one with higher FLOSS competency among the employees.

This ensures easier knowledge transfer in the future. Once the scope of the pilot project is set, the transition manager goes on → *establishing* and → *communicating the transition process*. After the process is completed, the same process replicates in other parts of the company and in other projects.

A.1.5 Define the transition timeline

Name	Define the transition timeline
Actor	Transition manager
Context	After → *defining the scope of the transition process*, the transition manager needs to define the transition timeline.
Problem	Transition is time bound. How should its timeline be best defined?
Solution	The transition manager must define a detailed timeline for the transition process. It should include the pilot project and the follow-up replication processes of the pilot project across the company. The timeline should be discussed with the affected projects and be approved by the transition board, where all the board members must come to a consensus over the defined timeline. When defining the timeline, consider: • time to establish the transition process • time to implement the transition process at the pilot project • time to replicate the process at other projects • time for the follow-up evaluation (optional).

A.1.6 Establish the transition process

Name	Establish the transition process
Actor	Transition manager, Transition board
Context	After → *defining the scope of the transition process* and → *defining the transition timeline*, the transition manager and the board must establish the transition process that is first implemented at a pilot project in the company, then replicated in other parts of the company.
Problem	The transition process is often the determinant of the transition's success. The specifics of the transition process depend on the company, on its use cases in terms of FLOSS use and on the capabilities of the employees. What are some common principles for establishing a smooth transition process?
Solution	The transition process outlines company's shift from unstructured, ad-hoc FLOSS governance to the structured and well-defined one. This handbook section covers how the company should get started with FLOSS governance. However, before the actual governance process, the pilot project and its team (and other projects consequently) must be exposed to the transition setup to prepare for the upcoming changes. Such a transition also ensures that the new governance approach and its motivation is clear to all the employees. During the transition the transition manager and board are established and introduced to the employees, which is the basis for implementation of all the other best practices. The transition process should follow these principles: • be clearly defined • be easy to follow without constant guidance by the board

- be scalable and replicable
- be assisted with tools that automate and/or ensure compliance with the process.

The transition process should include:

- outlining the motivation behind FLOSS governance
- clarifying the roles of the employees during the transition
- communicating the timeline and scope of the transition
- communicating the steps of the transition (product analysis and risk mitigation best practices) and their expected outcomes
- setting up new and structured procedures for decision making related to governance.

A.1.7 Communicate the transition process

Name	Communicate the transition process
Actor	Transition manager
Context	After → *establishing the transition process*, you must communicate it clearly first with the pilot project and then with the rest of the company. This must be done before → *implementing the transition process* to allow the affected teams to prepare for the transition.
Problem	Communicating the transition to FLOSS governance can be tricky, as the process is often complex for every single employee. How should it be communicated not be become too complicated for the employees?

Solution Best practice patterns of the handbook ensure easy communication of the transition process. The transition manager should present an overview of the transition to the employees, and point each employee group to the best practice patterns relevant only to them, while guiding them and serving as a central hub for discussions about the overarching issues. t's recommended that the transition manager meets the affected team (pilot project or other projects) at least three times during the transition:

- before the transition begins (to explain the motivation of the transition)
- mid-transition (to discuss the issues of various employee groups might have)
- after the transition (to document unresolved issues and to evaluate the process).

A.1.8 Implement the transition process

Name Implement the transition process

Actor Transition manager, Transition board

Context After → *establishing* and → *communicating the transition process*, you must implement the transition process at a pilot project or in the rest of the company.

Problem Implementing the transition process must be done gradually. How can this gradual implementation work in practice?

Solution You implement the established process that covers the essentials of the transition. For undefined questions, make case by case decisions and document them. The transition manager implements the process gradually, first introducing the overall process to the affected team, then demonstrating an example of identifying, documenting and clearing an open source component in a product.

Along the way, Q&A sessions and discussions between the affected team and the transition board ensure a smooth transition. The end goal of the transition process implementation is to ensure the affected team understands the changes introduced by this handbook, as well as the motivation behind these changes.

A.2 Transition Policy

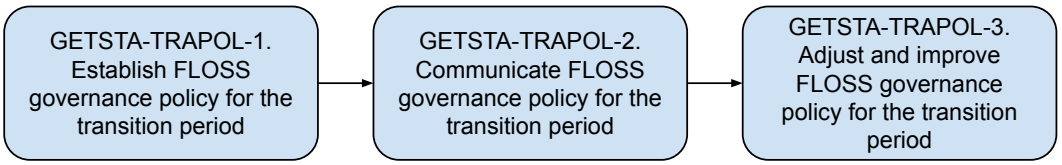


Figure A.3: Transition Policy Process Template

A.2.1 Establish FLOSS governance policy for the transition period

Name	Implement the transition process
Actor	Transition manager, Transition board
Context	Developers have been using open source components without any governance guidelines in the past. In parallel to introducing FLOSS governance at the company, you need to define and communicate the new rules for using open source components for the transition period, before → *implementing the transition process*.
Problem	How should you manage FLOSS governance during the transition period, before you have established a comprehensive FLOSS governance strategy?
Solution	In parallel to → *establishing the transition process*, establish FLOSS governance policy for the transition period that covers all the critical issues around the use of open source components in products, such as license compliance, bill of materials management, documentation, communication, etc.

The governance policy can be stored as a single document or divided into two separate documents:

- The first explains the intention of FLOSS use, defines the principles of using FLOSS in products. It outlines what kind of licenses, including their legal assessments and packages, are acceptable for use in commercial products, and pairs legal assessments with business use cases for each license.
- The second establishes a set of standards and tasks for the employees to follow to ensure compliance with FLOSS governance processes. This way, the policy can be implemented across the whole company under identical conditions. Also at larger companies, each division or department can adopt the policy with certain differences.

It is necessary to → *communicate FLOSS governance policy for the transition period* and to → *adjust and improve FLOSS governance policy for the transition period*.

A.2.2 Communicate FLOSS governance policy for the transition period

Name	Communicate FLOSS governance policy for the transition period
Actor	Transition manager
Context	After → *establishing FLOSS governance policy for the transition period*, it is necessary to make the policy accessible to the employees, so they can follow it before the introduction of a comprehensive FLOSS governance strategy.
Problem	What are the best channels to communicate the FLOSS governance policy for the transition period?

Solution Communicate FLOSS governance policy for the transition period using a variety of channels:

- the same channels that are currently used for the informal open source governance, such as mailing lists managed by key employees dealing with open source
- multi-purpose internal communication channels, such as intranet, wikis, forums, etc.
- internal video recordings and streaming channels, such as the introduction of the policy by a representative of the top management or by the transition manager

It is also recommended to communicate the details of the policy through employee trainings that include other topics around open source governance and the transition. To do this, consider the governance policy when → *designing employee trainings*. This ensures that your staff understands the significance and the principles of FLOSS governance and that the employees have a chance to ask clarifying questions about the policy.
The latter can be used to → *adjust and improve FLOSS governance policy for the transition period*.

A.2.3 Adjust and improve FLOSS governance policy for the transition period

---

Name Adjust and improve FLOSS governance policy for the transition period

Actor Transition manager, Transition board

Context	You → *established* and → *communicated FLOSS governance policy for the transition period*. You are getting clarification requests and questions from the employees.
Problem	What's the best way to address employee questions and clarification requests regarding FLOSS governance policy?
Solution	Iteratively collect and analyze the questions and requests you are getting. Identify the common misunderstandings and use them to adjust and improve FLOSS governance policy. Regularly post updates, clarifications, and FAQs to the affected employees via internal communication channels used to → *communicated FLOSS governance policy for the transition period*. It's important to update and maintain the policy by receiving feedback from the development teams to improve the processes. It is also essential to record violations to understand why certain development teams violate the process to minimize the issues in the future.

A.3 Product Analysis

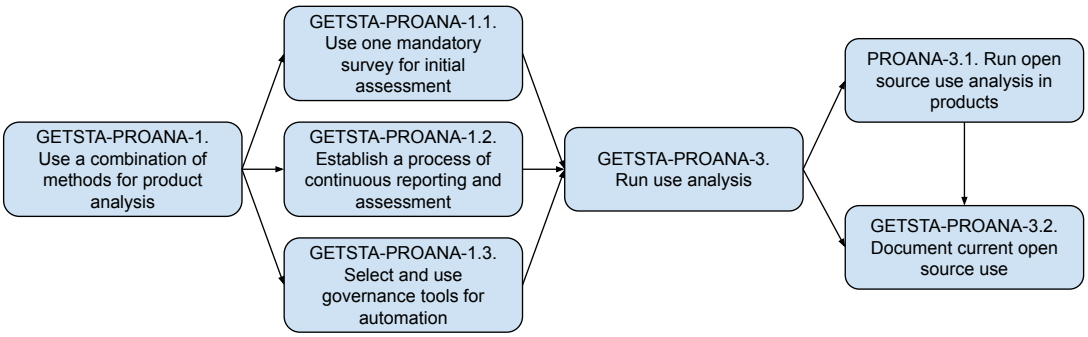


Figure A.4: Product Analysis Process Template 1

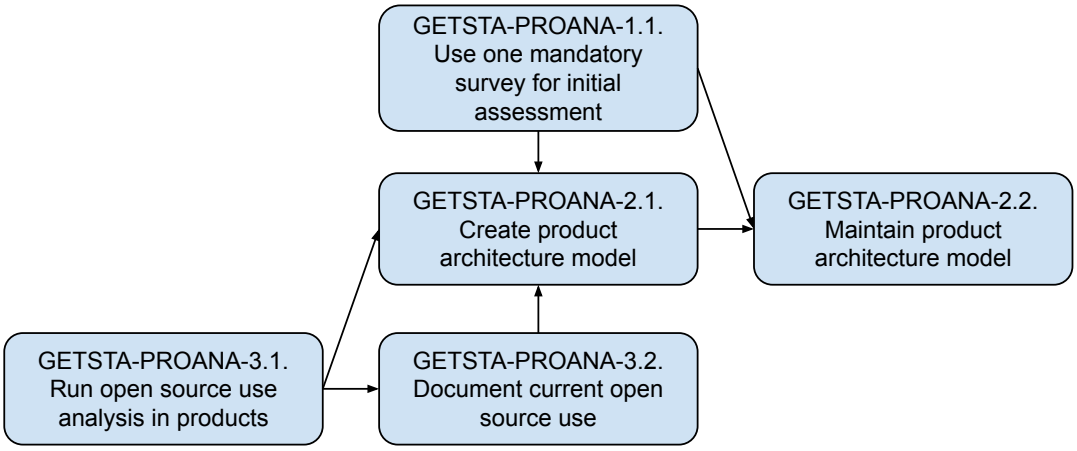


Figure A.5: Product Analysis Process Template 2

A.3.1 Use a combination of methods for product analysis

Name	Use a combination of methods for product analysis
Actor	Transition manager and / or Project architect
Context	After → *implementing the transition process*, you must review your current FLOSS use in products. You have been using open source in your products without defined or regulated governance processes or rules.
Problem	You must review your existing products and their open source components to → *ensure their license compliance* and to → *mitigate other risks*. What methods should you use for product analysis?
Solution	You should use a combination of different methods, including: ⇒ *using one mandatory survey for initial assessment* ⇒ *establish a process of continuous reporting and assessment* ⇒ *selecting and using the governance tools for automation.*

A.3.2 Use one mandatory survey for initial assessment

Name	Use one mandatory survey for initial assessment
Actor	Transition manager and / or Project architect
Context	You must collect all relevant information on previous FLOSS use in products. While some data can be found directly in the source code, there is some tacit understanding of FLOSS governance and implicit rules around it. You are considering to → *use a combination of methods for product analysis*.

Problem	The tacit understanding of FLOSS governance, the implicit rules around FLOSS governance and any data on current use of FLOSS components must be elicited. How should you do this?
Solution	Design and run one mandatory survey for initial assessment of FLOSS use in products. This qualitative survey (questionnaire) has the goal of identifying any previously undocumented use of open source in company's products. It also sets out to understand the current perception of the developers and managers on open source governance related decisions they have made in the past. The target of the survey are project's developers and (engineering) management, as well as other employees who made decisions on use of open source in products (e.g. lawyer responsible for open source, IT department, procurement office etc.). The survey should not be overwhelming. It must be well structured. The results should enable easy documentation of initial assessment findings.

A.3.3 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 use 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 a 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 licence 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.

A.3.4 Select and use governance tools for automation

Name Select and use governance tools for automation

Actor Project architect

Context	In parallel to → *establishing a process of continuous reporting and assessment*, you should try to automate as much of the reporting and assessment as possible. This ensures higher efficiency and minimal distraction for developers.
Problem	Which aspects of open source governance during the transition should be automated and how does one select the right tools?
Solution	The main focus of automation during transition should be reviewing the source code of the existing products. This includes scanning company's software for open source licenses to be used for license compliance clearance. This also includes scanning for copyright notices, export restrictions and security vulnerabilities. The selected tools must provide an easy integration with the development environment, provide adequate documentation, and enable further functionality, such as support for bill of materials management automation, supply chain management, etc. To select the right tools, study your company's use cases, FLOSS governance needs, and scale of open source usage. You should consider both open source tools and proprietary tools for open source governance. If one tool is already used in the company, you should consider extending its use to ensure the centralization FLOSS governance across the whole company.

A.3.5 Product architecture model

A.3.6 Create product architecture model

Name	Create product architecture model
Actor	Transition manager and / or Project architect

Context	You already → *used one mandatory survey for initial assessment* of FLOSS components in your products, → *established a process of continuous reporting and assessment*, and → *selected the governance tools for automation*. You are now ready to set up a structured and integrated product architecture model.
Problem	Products keep changing, new open source components are being added and modified. You need to keep track of the ongoing changes in product architecture and to maintain structured metadata (e.g. license information, reuse information, export restrictions etc.) on open source components.
Solution	Create product architecture model to set up and maintain a structured and formalized view of software components used in your products. Define open source component-specific properties within the model to allow collection, tracking, maintenance, and monitoring of metadata including open source license information, export restrictions, known security vulnerabilities, software dependencies, etc. If possible, create a product architecture model integrated into the build process or continuous development process to ensure higher automation.

A.3.7 Maintain product architecture model

Name	Maintain product architecture model
Actor	Project architect
Context	You previously → *created a product architecture model*. Short delivery cycles, including continuous deployment, require that you can generate (and provide) compliance artifacts (among other things) at any time.
Problem	Creating a product architecture model from scratch takes longer than may be acceptable with short delivery cycles. How to make sure you can provide compliance artifacts on demand?

Solution	Whenever you make a change to the component architecture of the product, update the product architecture model. Do not forget to version the product architecture model. This way, the model will always be up to date.

A.3.8 Use Analysis

A.3.9 Run open source use analysis in products

Name	Run open source use analysis in products
Actor	Product architect, Developers
Context	Before → *creating a product architecture model*, you need to analyze what open source software is in current use in your company's products.
Problem	To have a structured view of open source components used in the company's products, there is a need for an initial review of current open source use. This initial review cannot be totally automated as different products have different architectures, and there is no common method of denoting open source components and their dependencies in products. How can the initial analysis be conducted?
Solution	To run open source use analysis in products, you must → *use a combination of methods for product analysis*. Talk to different product development teams and identify their current practice or conventions of denoting open source use. If there is an enforced formal or informal convention, use it to automate the initial product analysis. If not, use software project management and comprehension tools or software component management tools to automate as much of the initial product review as possible. Afterwards, survey technical managers or team leads to gather information on the open source components used by developers in their respective teams. The results open source use analysis in products will be used to → *document current open source use*.

A.3.10 Document current open source use

Name	Document current open source use
Actor	Product architect
Context	After → *running open source use analysis in products*, you need to document the results of your analysis in a structured format, which will then be used to → *create product architecture model* and to document the current use there.
Problem	The initial analysis of open source use in products results in semi-structured information on current open source use. How can you transfer this semi-structured information into structured information while identifying the key properties of them to be created product architecture model?
Solution	Based on the semi-structured information gathered while → *running open source use analysis in products* you need to standardize the open source use related information using an established data exchange format (for example SPDX format). When documenting your current use, keep in mind that this documentation should be easily transferable into the product architecture model, where it will be placed and → *maintained*.

A.4 IP-at-Risk Analysis

A.4.1 License Compliance Analysis

A.4.2 Develop standard license interpretation

Name	Develop standard license interpretation
Actor	Lawyer / Legal counsel
Context	Software developers need legal advice on open source licenses before using given components in company's products in order to ensure legally compliant use of open source software. There are about 20 prominent and widely used open source licenses with strong communities using them. These standard licenses are approved by the Open Source Initiative.
Problem	Software developers keep asking the legal council about open source license interpretation for each software component with a new license. This is redundant and time-consuming work for both lawyers and software developers.
Solution	Legal counsel needs to develop standard open source license interpretation agreed upon internally, taking into account existing ambiguities and potential risks of each open source license. The interpretations for 20 most prominent and popular open source licenses need to be documented and communicated clearly with software developers and technical managers working in product development. Once the standard interpretation of licenses is developed company employees must → *use standard license interpretation*, when possible.

A.4.3 Use standard license interpretation

Name	Use standard license interpretation
Actor	Developers
Context	Software developers need legal advice on open source licenses before using given components in company's products in order to ensure legally compliant use of open source software. Your company's lawyer → *developed standard license interpretation* and shared them with developers across the company.
Problem	Who should use the standard license interpretations and how?
Solution	Developers must use and follow company's standard license interpretations when adding open source components into company's products. Developers should be introduced to the standard license interpretation of the major licenses (GPLv2, GPLv3, LGPL, AGPL, MIT, BSD, Modified BSD, etc.) during the → *provided employee training*. When using open source code (either directly from FLOSS communities or as part of supplied software), developers should consider the license interpretation in a given business case (e.g. using GPLv3 can be acceptable in one use case and not acceptable in another) generally outlined in company's → *established FLOSS governance policy for the transition period*. For the special cases that are not described in the governance policy, developers must consult the transition board or the transition manager, who then review and document their case by case decisions as part of the → *implemented transition process*. The transition manager must use this documentation to → *create license/use case pairs*.

A.4.4 Create license/use case pairs

Name	Create license/use case pairs
Actor	Transition manager
Context	Your company → *developed standard license interpretation* and you are → *using standard license interpretation*. Developers are also consulting company's → *established FLOSS governance policy for the transition period*, and are contacting the transition board or the transition manager for case by case review of special cases of FLOSS use.
Problem	What's the best way to document the case by case decisions on special cases of FLOSS use, reviewed by the transition board or the transition manager?
Solution	In one centrally available document, create license/use case pairs to document the case by case decisions on special cases of FLOSS use. This document should include all the major licenses and company's detailed approach to their use in different business contexts or use cases. For example, it can be acceptable to use a copyleft license for certain (non-differentiating) products, while it might be unacceptable in other cases such as for company's main products (with competitive advantage). Such license/use case pairs should be well structured and documented. In case of a new decision on a special license/use case pair by the transition board, this document must be updated by the transition manager. Developers must consult the document before contacting the transition board or the transition manager with a new review request, because they might be able to find their answer for a specific license/use case pair in the document. Having such a document improves performance and reduces unnecessary redundancy.

A.4.5 Analyze risk exposure of using an open source component

Name	Analyze risk exposure of using an open source component
Actor	Engineering manager and / or Product architect
Context	After you completed → *Use Analysis* and → *License Compliance Analysis*, you are analysing the found inconsistencies and / or compliance issues. You need to analyze risk exposure before deciding what to do about the identified issues.
Problem	What's the best way to analyze risk exposure of using an open source component?
Solution	Analyze risk exposure of using open source components by employing tools and by creating and using risk assessment matrix. Aim to automate risk analysis for the identified FLOSS use by → *selecting and using the governance tools for automation*. At the same time follow the → *established FLOSS governance policy for the transition period* for general principles on FLOSS use related risk analysis. For detailed risk assessment, establish a matrix of risk levels and their criticality to the company. Use and update this matrix over time, sharing it with the developers and other stakeholders in the company. To assess risk exposure (and its level in the matrix) consider where the observed open source component is used: • in a consumer-facing product • in a B2B product • for a demo product (not for sale) • internally as development tools • internally as part of R&D

Define your company's level for "minor" or "acceptable" risks upon consultation with the transition board. Document and share these levels in the → *adjusted and improved FLOSS governance policy for the transition period*. If the risk exposure is high, inform developers about the need to take steps towards → *IP Risk Mitigation*.

A.4.6 IP Risk Mitigation

A.4.7 Replace problematic components

Name	Replace problematic components
Actor	Developers and / or Product architect
Context	The transition manager has completed → *Use Analysis*, → *License Compliance Analysis* and → *Risk Exposure Analysis*. The transition manager assesses the risk exposure of using some open source components to be high.
Problem	What can be done to mitigate the IP risk, if there is no workaround?
Solution	If the observed open source component has a license/use case pair that is not acceptable for the company and if there is no workaround (e.g. contacting the original copyright owner and considering dual licensing for copyleft licenses), you must replace problematic components as soon as the risk exposure has been discovered. Before replacing the component, you must identify a viable and functional alternative that will not affect the product. You must consult with the engineering manager about the costs associated with the replacement. If you cannot replace the component, you must inform the transition manager, who will document this as an existing risk and investigate other action plans.

A.4.8 Decouple problematic components

Name	Decouple problematic components
Actor	Developers and / or Product architect
Context	The transition manager has completed → *Use Analysis*, → *License Compliance Analysis* and → *Risk Exposure Analysis*. The transition manager assesses the risk exposure of using some open source components to be high.
Problem	What can be done to mitigate the IP risk, if there is a workaround?
Solution	If the observed open source component has a license/use case pair that is not acceptable for the company, but if there is a workaround (e.g. by linking the component the to main product code or by changing the type of code linking), decouple problematic open source components by ensuring license compliance and as a result by complying with company's accepted license/use case pair.

A.4.9 Require bill of materials for supplied code by third-party post-factum

Name	Require bill of materials for supplied code by third-party post-factum
Actor	Product architect and / or Procurement office
Context	The transition manager has completed → *Use Analysis*, → *License Compliance Analysis* and → *Risk Exposure Analysis*. The transition manager detects high risk exposure of using some open source components that came through supplied code by a third-party, but were not detected earlier.
Problem	What can be done to mitigate the IP risk caused by non-compliant code supplied by a third-party?

Solution	Post-factum require bill of materials for supplied code by a third-party to check if your risk analysis was correct and if the component in question is actually non-compliant (e.g. with open source component's license). If possible, require the bill of materials in SPDX format that can be easily read by most governance tools, as well as component management tools. If the supplier supplied non-compliant code, request a replacement or decoupling of the component and / or financial reimbursement, if legally possible.

A.4.10 Run random audits to identify previously undetected or missed open source components and their metadata

Name	Run random audits to identify previously undetected or missed open source components and their metadata
Actor	Transition manager and / or Quality engineer
Context	Companies must ensure that their commercial products are not subject to intellectual property risks because of the open source software used.
Problem	After → *using one mandatory survey for initial assessment*, how should you check your use of open source?
Solution	After → *using one mandatory survey for initial assessment*, run irregular audits to identify previously undetected or missed open source components and their metadata. It is necessary to have randomly scheduled audits for projects or products. The audits should be conducted by quality engineers and / or transition manager. During the audits, it is necessary to check what was presented, reviewed and approved in the documented product architecture model, and what wasn't.

After the audit, the newly identified open source components must be documented in the product architecture model and go through → *Risk Exposure Analysis*.

A.4.11 Analyze security risk of using an open source component

Name	Analyze security risk of using an open source component
Actor	Security manager and /or IT officer and / or Product architect
Context	In parallel to finding inconsistencies and / or compliance issues, you want to check your open source component for potential security risks.
Problem	What's the best way to analyze the security risk of using an open source component?
Solution	In parallel to conducting → *Use Analysis* and → *License Compliance Analysis*, you can analyze security risk of using an open source component. You should automate this process by using tools that support for open source governance and security management. Before the analysis you can consult with: • security officer • lawyer / legal counsel • engineering manager • IT department. You can consider including security-related questions in the → *one mandatory survey for initial assessment*.

A.5 Communication and Capabilities

A.5.1 Establish communication channels for open source governance handbook

---

Name	Establish communication channels for open source governance handbook
Actor	Transition manager
Context	For any communication related to open source governance and this handbook, you need to establish communication channels that will be the information bridge between the transition board and the affected employees.
Problem	How should communication about open source governance be organized?
Solution	Communicate FLOSS governance policy for the transition period using a variety of channels: • the same channels that are currently used for the informal open source governance, such as mailing lists managed by key employees dealing with open source • multi-purpose internal communication channels, such as intranet, wikis, forums, etc. • internal video recordings and streaming channels, such as the introduction of the policy by a representative of the top management or by the transition manager. It is also recommended to communicate the details of the policy through employee trainings that include other topics around open source governance and the transition. To do this → *design employee trainings* and → *provide employee trainings*.

	Use these principles to → *communicate the transition process* and to → *communicate FLOSS governance policy for the transition period.*

A.5.2 Assess open source governance capabilities among developers and engineering manager

Name	Assess open source governance capabilities among developers and engineering manager
Actor	Transition manager
Context	For the transition phase, you need to identify the employees who already have a good understanding of open source governance based on their education or previous experiences, in order to use these capabilities for transition tasks.
Problem	How can you find out who are the employees with a good understanding of open source software?
Solution	Assess open source governance capabilities among developers and engineering manager by adding capability assessment questions in the → *one mandatory survey for initial assessment*. Engage the employees with good open source governance capabilities in transition tasks in their teams / divisions. Reward them for their contributions. Enable them to share their knowledge with their fellow employees.

A.5.3 Develop FLOSS governance and compliance capabilities at the central legal department

Name	Develop FLOSS governance and compliance capabilities at the central legal department

Actor	Transition manager and Lawyer / Legal counsel
Context	The legal department is of high importance to FLOSS governance, but very few have experts on open source.
Problem	How should you address the issue lacking FLOSS capabilities at the legal department?
Solution	Develop FLOSS governance and compliance capabilities at the central legal department in order to have one point of contact for legal issues related to open source. If you can, hire a lawyer who is already competent in dealing with the commercial use of open source. Alternatively, build FLOSS related legal capabilities in house by sending your lawyer to conferences and workshops and by encouraging their development in this field.

A.5.4 Design employee training

Name	Design employee training
Actor	Transition manager
Context	Employee training is intended to increase the awareness of the commercial use of open source and of FLOSS governance in a company. Also, training gives a common understanding of strategic and technical implications, and a common mindset on FLOSS governance.
Problem	How does one build a common understanding of FLOSS governance issues and risks? How to increase the awareness of FLOSS compliance and governance among developers and others?
Solution	Employee groups who will be involved in the training program should be initially identified (e.g. developers and architects, or team managers as well as legal team, supply chain team etc.).

All the training materials including necessary forms and documentation should be designed with the target audience in mind. For example, developers and architects receive a web-based training and attend company-wide talks focused specifically on open source governance and compliance. However, company executives receive only brief information about the FLOSS governance policy. Design easy to understand but useful training. Consider repeating the key information for better retainment. After designing it, → *provide employee training* to the selected target groups.

A.5.5 Provide employee training

Name	Provide employee training
Actor	Transition manager
Context	Employee training is intended to increase the awareness of the commercial use of open source and of FLOSS governance in a company. Most employees need training during the transition to understand and to follow the best practices of this handbook.
Problem	How should you provide employee training?
Solution	After → *designing employee training*, provide it to the selected target groups. Provide regular trainings once or twice a year (more frequently, if needed) to team managers and developers. Provide trainings on request in the early phase of the transition for discussions and Q&A sessions. Make sure that training materials that can be easily accessible at any time and that they are manageable and can be comfortably navigated.

B

FLOSS Governance Handbook – Selected Practices for Supply Chain Management

Appendix B presents an excerpt from the open source governance handbook's section on supply chain management. The handbook presents the findings of our research cast as a collection of interconnected patterns. This presentation format enables higher applicability of the research results. This appendix also includes process templates – example workflows connecting various best

practices on supply chain management, which further improves the practical applicability of our results. See Figure B.1 and Figure B.2 for process templates.

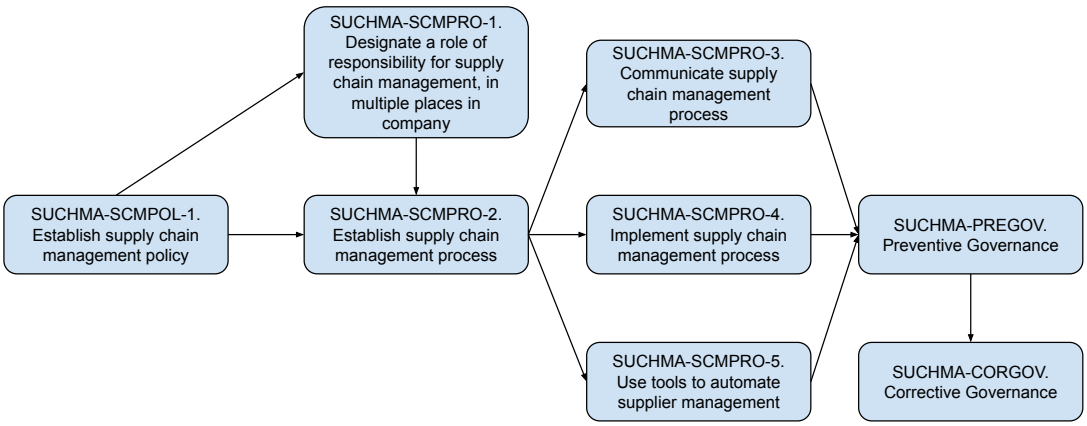


Figure B.1: Supply Chain Management Process Template 1

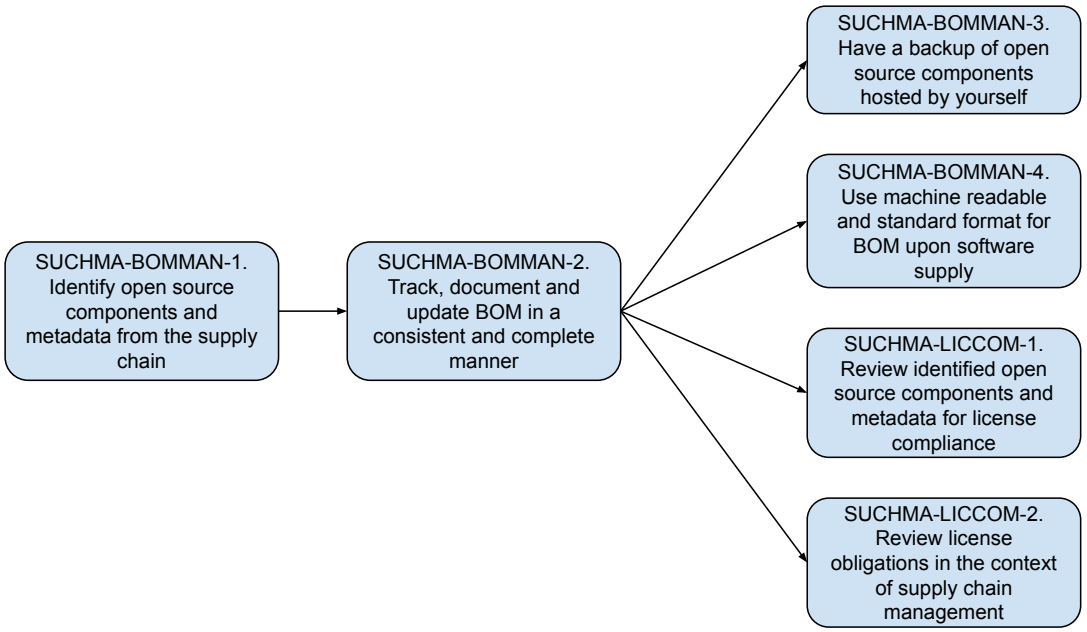


Figure B.2: Supply Chain Management Process Template 2

B.1 Supply Chain Management Policy

B.1.1 Establish supply chain management policy

Name	Establish supply chain management policy
Actor	OSPO (Open Source Program Office)
Context	After → *defining goals of governance* and → *establishing an open source program*, you defined roles, responsibilities, and policies to address various aspects of open source governance in an abstract manner. Now you need to define the specific policy for managing software suppliers and open source software that you get through them either directly or indirectly. Having such a policy will help mitigate license non-compliance risks associated with the use of open source components supplied to you.
Problem	Without such a supply chain management (SCM) policy you do not have a systematic way of addressing open source governance and compliance within your software supply chains. Unlike the open source components introduced by a company's own developers, supplied code can have unidentified open source components that could pose compliance risks. One of the reasons is that most software suppliers supply only a binary and not the source code. While the source code can be easily scanned for its open source components and their metadata such as licenses, binaries are harder to scan reliably and require specialized tools. Identifying open source components that have been used by the suppliers of your supplier on the next tiers of the supply chain are even harder and imprecise. Different suppliers have different levels of understanding and awareness in regards to open source licenses, compliance, and governance. This results in different compliance risk levels associated with different suppliers.

The critical dependence on certain suppliers increases the complexity of such an ungoverned use of open source components supplied as part of purchased software components for used in your final products. How can you ensure open source governance and compliance while managing the complexities of software supply changes?

Solution In parallel to → *establishing the supply chain management process*, establish supply chain management policy that the company's approach to managing open source components acquired through software suppliers and not through the internal → *component approval process* supported by the Open Source Program Office. The supplier management policy presents a set of principles that guide a company through the FLOSS governance in its supply chain to help identify all the open source components used in the supplied code on all tiers of the supply chain and their metadata. The policy often has two logical parts: the intention of the policy explaining the Open Source Program Office's motivation for having this policy, and guidelines and best practices that are later operationalized through the → *established the supply chain management process*. In particular, the aspects covered by the policy include but are not limited to:

- company goals for supplier management
- metrics for efficient supplier management
- principles for managing open source components in the supplied code
- recommendations for automating supplier management through tools
- rules for suppliers that use open source components
- guidelines for managing supplied open source components
- supplier management integration with other aspects of open source governance, including component approval, component tracking, and license compliance
- recommended best practices for managing supplied open source components.

Select industry best practices for supplier management are presented in this handbook, such as:

⇒ *Assess open source governance and compliance awareness and maturity*

⇒ *Request supplier certification or self-certification*

⇒ *Design supplier contracts with open source governance aspects in mind*

⇒ *Audit your supply chain*

⇒ *Don't run your supplier out of business*

⇒ *Get the source code (before changing the supplier)*

⇒ *Identify Open Source Components and Metadata from the Supply Chain*

⇒ *Have a Backup of Open Source Components Hosted by Yourself*

⇒ *Use Machine-readable Bill of Materials (BOM) upon Software Supply*

⇒ *Use Standard Format for BOM upon Software Supply*

⇒ *Review License Obligations in the Context of SCM*

⇒ *Use tools to automate supplier management.*

The supplier management policy helps the Open Source Program Office define a consistent approach to the issue that can be systematically documented, implemented and communicated across the organization. It should evolve and can be modified by the Open Source Program. It should be easy to read and to the point with appropriate references to other parts of this handbook. The policy should be created in collaboration with the engineers.

If the lawyers create it, it will be comprehensive and well written, but few people will ever read or follow it, because of its complexity and language. The policy should be related directly to the day to day tasks of software development and solve problems that engineers and managers face in their work.
The policy should address the principles and company approach to the key topics of supplier management:

- Supply chain management process
- Preventive supply chain management governance
- Corrective supply chain management governance
- Bill of materials management
- License compliance for supply chain

The specific guidelines and best practices for these topics are presented in the other parts of this section. The policy is operationalized through the → *supply chain management process* that defines the steps that developers and other roles need to follow in order to efficiently manage a company's supply chain and the related open source component in their products or projects. Once the policy is established, it is necessary to → *communicate supply chain management policy* and to → *adjust and improve supply chain management policy*.

B.1.2 Communicate supply chain management policy

Name	Communicate supply chain management policy
Actor	OSPO (Open Source Program Office)

Context After → *establishing supply chain management policy*, it is necessary to make the policy accessible to the employees, so they can follow it and ask clarification questions to the OSPO.

Problem How should you communicate the supply chain management policy?

Solution Once the policy is written, you need to simplify it highlighting the key aspects for different roles in the company (engineers, management, lawyers, procurement etc.) and share with all the affected stakeholders in the company and among suppliers. Supplier management policy has the following recipient stakeholders:

- internal
  - designated role of responsibility for supply chain management
  - software developers
  - technical management
  - business management
  - lawyers
  - procurement department
  - IT department
- external
  - tier 1 suppliers
  - tier 2 and other suppliers
  - supplier associations

The internal stakeholders should follow the policy in their day to day tasks, alongside with the → *supply chain management process*.

Developers follow the policy to document and update the bill of materials, technical and business managers ensure that the supplied code corresponds to company's requirements, lawyers and procurement department design and manage supplier contracts with the governance related issues in mind, IT department install and maintains the supplied tools and components. In all these tasks employees should follow the policy for questions not explicitly defined in the → *supply chain management process*. They need to provide their questions and feedback to the OSPO to → *adjust and improve supply chain management policy*.

The external stakeholders should follow the policy to meet a company's governance and compliance requirements for software suppliers and to ensure strategic alignment across the supply chain. It is more efficient to communicate open source governance and compliance issues in the beginning of the supplier relationship (before and during contract negotiation) than after the delivery of the supplied code. In case of poor governance and compliance by certain suppliers, the supplied code can affect the whole product where it is integrated and thus create risks that are costly to address post factum.

Communicate the supply chain management policy using the → *regular channels that OSPO uses for open source governance and compliance related internal communication*. The main recipients of OSPO's communication should be the employees in different development teams that occupy the → *designated role of responsibility for supply chain management*. For some parts of the policy, these employees should forward the specifics or changes of the policy to individual developers and other internal or external stakeholders. At the same time, these designated employees are responsible for collecting the commonly asked employee questions and forwarding them to the OSPO, as well as collecting and sharing OSPO's answers with colleagues.

B.1.3 Adjust and improve supply chain management policy

Name	Adjust and improve supply chain management policy
Actor	OSPO (Open Source Program Office)
Context	You → *established* and → *communicated supply chain management policy*. You are now getting clarification requests, questions and feedback on the policy from the employees.
Problem	How should you address employee feedback regarding supply chain management policy?
Solution	In the early phase of policy, rollout anticipate frequent questions by employees and external suppliers which can result in changes, clarifications, and adjustments to the policy. There is no one-size-fits-all policy for supply chain management and open source governance. You will have to specify various principles of the policy over time, as well as operationalize them in the → supply chain management process. Iteratively collect and analyze the questions and requests you are getting. Identify the common misunderstandings and use them to adjust and improve the policy. Instruct the → *designated role of responsibility for supply chain management* to regularly post updates, clarifications, and FAQs to the affected employees via internal communication channels used to → *communicate supply chain management policy*. It's important to update and maintain the policy by receiving feedback from the development teams and others to improve the processes. It is also essential to record violations to understand why certain stakeholders violate the policy to address and minimize similar issues in the future.

B.2 Supply Chain Management Process

B.2.1 Designate a role of responsibility for supply chain management, in multiple places in company

---

Name	Designate a role of responsibility for supply chain management, in multiple places in company
Actor	OSPO (Open Source Program Office)
Context	Virtually all software products contain open source components that are either added by your software developers directly or come into your company through the software supply chain. You use a → *component approval process* to systematically manage the components added by the developers, but software supply chains are not managed in terms of open source governance and compliance.
Problem	Many internal and external stakeholders are involved in software supplier management, but none has the explicit responsibility of dealing with FLOSS related issues. Who should ensure and manage the efficient supply chain management across your company?
Solution	The OSPO is responsible for open source governance in general and for open source in software supply chains in particular. However, a centralized body like the OSPO needs local support in different divisions and teams of the company to coordinate the internal and external stakeholders. OSPO's role is to: ⇒ *Establishing supply chain management policy* ⇒ *Communicate supply chain management policy*

⇒ *Adjust and improve supply chain management policy*

⇒ *Establish supply chain management process*

⇒ *Communicating supply chain management process.*

However, OSPO does not have the resources to locally → *implement supply chain management process* in every project and division of the company. To address this, OSPO should create a role of responsibility for supplier management which can be delegated to employees within different projects in the company.
Depending on the company, recommended candidates for such a role include, but are not limited to:

- for internal supply chain management
  - (software) project managers for internal
  - (software) product managers
  - technical product managers
  - senior developers
  - IT manager
- for external supply chain management
  - procurement manager
  - IT manager.

Employees responsible for supply chain management should spend only a small part of their work time in component in this role. However, this role can be combined with other similar responsibilities in other areas of open source governance, such as the → *designated role of responsibility for the component repository*.

Employees responsible for supply chain management support their teams and divisions to:

⇒ *Assess open source governance and compliance awareness and maturity of a supplier*

⇒ *Assess governance maturity of a supplier*

⇒ *Request supplier certification or self-certification*

⇒ *Audit your supply chain*

⇒ *Trigger supplier contract clauses and get the supplier to take care of the issue*

⇒ *Identify Open Source Components and Metadata from the Supply Chain*

⇒ *Document BOM in a Consistent and Complete Manner.*

Once the employees are designated, OSPO should present their new tasks and should introduce these employees to their teams in the new capacity. Designated employees should ensure the communication between the teams and OSPO, as well as feedback and updates from both sides. Employees tasked with external supplier management should ensure the communication between suppliers and OSPO.

B.2.2 Establish supply chain management process

Name	Establish supply chain management process
Actor	OSPO (Open Source Program Office)

Context After → *establishing supply chain management policy* in accordance with the company's → *defined goals of governance*, you must now define how exactly should the employees manage suppliers for open source governance.

Problem You → *established supply chain management policy* and → *communicated supply chain management policy*.

However, the supplier management policy is too broad to apply operationally. It focuses on company's principles for supplier management in regard to open source components, but leaves out the operational aspects of this management. How should you operationalize the policy in the day to day life of the company, while making sure it is widely accepted and used?

Solution You need to establish a process that guides your employees in different roles through the operational aspects of governed supplier management. As using supplied software components in products is virtually inevitable and necessary for efficient development, this process will ensure that no unwanted open source components will end up in your products through the supplied code. The centrally defined and unified process has a number of benefits including, but not limited to:

- point of reference for new developers or managers who need to manage suppliers and their supplied open source components
- a centralized approach for bill of materials management and maintenance with all the used open source component across the organization
- consistent and up-to-date bill of materials for compliance and release management
- search of the open source components in bill of materials and their metadata.

Similar to the → established transition process from the Getting Started section of this handbook, the component reuse process should follow these principles:

- be clearly defined
- be easy to follow without constant guidance by the OSPO (OSPO should handle the exceptions on a case by case basis)
- be scalable and replicable
- be assisted with tools that automate and/or ensure compliance with the process.

First, you need to establish preparatory steps for the supplier management process. For this, you need to:

⇒ *Established supply chain management policy*

⇒ *Communicated supply chain management policy*

⇒ *Adjust and improve supply chain management policy*

⇒ *Designate a role of responsibility for supply chain management, in multiple places in the company*

Then, you need to define and perform:

⇒ *Preventive governance*

⇒ *Corrective governance*

⇒ *Bill of materials management*

⇒ *License compliance for supply chain*

The specifics of these measures is described in the practice on the → *implementation of the process*.

B.2.3 Communicate supply chain management process

Name	Communicate supply chain management process
Actor	OSPO (Open Source Program Office)
Context	After → *establishing a supply chain management process*, you must communicate it clearly with the → *employees designated a role of responsibility for supply chain management* and other affected employees who will be using the process.
Problem	The supply chain management process must be communicated to the affected employees across the company. The process can be complex and not all parts of the process are relevant for everyone. How should you communicate the supply chain management process before → *implementing the supply chain management process*?
Solution	You should present the overview of the supply chain management process to the affected employees, and point each employee group / role to the best practice patterns relevant only to them, while guiding them and serving as a central hub for discussions about the overarching issues. Your direct communication should be with the → *employees designated a role of responsibility for supply chain management*. After that, it's recommended that they meet their teams to process and to clarify the steps, focusing on: • preparatory steps for the supply chain management process • preventive governance measures • corrective governance measures • bill of materials management • license compliance for supply chain.

For efficient communication, use the → *regular channels that the OSPO uses for open source governance and compliance related internal communication*.

B.2.4 Implement supply chain management process

Name	Implement supply chain management process
Actor	OSPO (Open Source Program Office), Supply chain management responsibles
Context	After → *establishing a supply chain management process* and → *communicating it*, you must implement the process across the company.
Problem	Implementing a large-scale process across the company has its challenges. How should you implement an efficient process?
Solution	Gradually implement the supplier management process. Start by introducing preventive governance measures. Preventive governance ensures that potential suppliers have a high degree of open source governance and compliance awareness, and thus are unlikely to supply non-compliant code. Preventive governance includes the following best practices: ⇒ *Choose the right supplier* ⇒ *Assess open source governance and compliance awareness and maturity* ⇒ *Request supplier certification or self-certification* ⇒ *Design supplier contracts with open source governance aspects in mind* Corrective governance ensures that after the code supply any governance and compliance issues are identified are corrected and addresses, mitigating the risks caused by suppliers.

Even after preventive governance there is a potential risk in terms of FLOSS governance and compliance, so the following best practices should be applied:

⇒ *Audit your supply chain*

- *Enable regular audits*
- *Enable surprise audits*

⇒ *Mitigate identified risks*

- *Assess risks in accordance to the supply chain management policy*
- *Trigger supplier contract clauses and get the supplier to take care of the issue*
- *Don't run your supplier out of business*

In parallel to preventive and corrective governance measures, focus on bill of materials management and on license compliance for supply chain. For bill of materials management, follow the best practices of the handbook to:

⇒ Identify open source components and metadata from the supply chain

⇒ *Track, document and update BOM in a consistent and complete manner*

⇒ *Have a backup of open source components hosted by yourself*

⇒ *Use machine-readable and standard format for BOM upon software supply*

For license compliance for supply chain, follow the best practices of the handbook to:

⇒ *Review identified open source components and metadata*

⇒ *Review license obligations in the context of SCM*

⇒ *Review copyright notices in the context of SCM*

⇒ Review security vulnerabilities in the context of SCM.

Depending on your specific needs you should modify the essential process to include more steps that would guide the employees in reusing open source components. Finally, the OSPO should handle the exceptions that deviate from the implemented process on a case by case basis, while considering process optimization and continuous improvement.

B.2.5 Use tools to automate supplier management

Name	Use tools to automate supplier management
Actor	OSPO (Open Source Program Office), Supply chain management responsibles
Context	Companies often have hundreds of suppliers. Each supplier provides multiple software deliveries with multiple open source components in each, as well as open source software provided by tier 2 and other suppliers.
Problem	It is not possible to manually deal with the complexity of software supply chains. How can companies deal with this issue in parallel to → *implementing the supply chain management process*.
Solution	Some aspects of supplier management can and should be performed using tools. Tools should be used for preventive governance when you choose a supplier to build and maintain a database of suppliers and their → *assessed maturity of open source governance and compliance*. Other tools can be used for → governance awareness self-certification by suppliers. Lawyers can use tools that assist in → *designing supplier contracts with open source governance aspects in mind*.

Tools should also be used in corrective governance and license compliance in supply chains. This includes open source code and license scanning tools that automate → *audits of the supplied software*. In bill of materials management tools should be used for → *tracking, documenting and updating bill of materials*, and to → *host a backup of the supplied FLOSS components*. Tools should be used to integrate supplier management process with other processes and artefacts of this handbook in component approval, component reuse etc.

B.3 Preventive Governance

B.3.1 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*.

B.3.2 Assess open source governance and compliance awareness and maturity

Name	Assess open source governance and compliance awareness and maturity
Actor	Supply chain management responsibles, IT department, Procurement department
Context	Companies use supplied software components in their products, but choosing the wrong supplier in terms of open source governance and compliance maturity can cause potential financial and legal risks.
Problem	To avoid governance and compliance risks caused by your supply chain you → *choose the right supplier*. How can you do that if you have many suppliers?
Solution	You need to assess open source governance and compliance awareness and maturity of the potential suppliers to avoid potential risks of license violations or other governance issues. Companies can demonstrate their knowledge and experience in FLOSS governance by demonstrating their internal governance process, by providing detailed bill of materials with highlighted data on the used open source components and their metadata, as well as through → *governance and compliance certification*. Make sure to add a clause about governance awareness and maturity assessment, when → *designing supplier contracts*. Document the assessment results for the new suppliers in a centralized company-wide database that can be used by other divisions, which will help make decisions about contracting certain suppliers. A systematic and consistent awareness and maturity assessment is the best way to prevent future issues with open source license compliance. This can save financial and legal resources that would otherwise be spent on corrective governance if issues are identified regarding suppliers' use of certain open source components as the final responsibility for all the components lies with the final client (e.g. OEM) in the end of the supply chain.

B.3.3 Request supplier certification or self-certification

Name	Request supplier certification or self-certification
Actor	Supply chain management responsibles, IT department, Procurement department
Context	You are → *assessing open source governance and compliance awareness and maturity* of the potential suppliers before signing a contract with them.
Problem	How should you assess a supplier's governance and compliance awareness and maturity on a large scale with a large number of suppliers?
Solution	Most companies have many software suppliers and have higher negotiation power compared to the suppliers. This enables companies to ask for certain certifications by suppliers. Such a certification requirement should also be applied to governance and compliance awareness assessment. The certification would cover all the major aspects of open source governance and compliance, including processes for components approval, component integration and reuse, compliance and release management, as well as supplier management performed in turn by the suppliers. An example of a currently available self-certification is called OpenChain, which is an effort by multiple companies to ensure preventive governance in their supply chain rather than performing costly and complex corrective measures. The advantage of self-certification is the easy access to certification by suppliers. Such a certification requirement can be optional or mandatory.

B.3.4 Design supplier contracts with open source governance aspects in mind

Name	Design supplier contracts with open source governance aspects in mind

Actor Supply chain management responsibles, Lawyer / legal counsel, Procurement department

Context You are → *assessing open source governance and compliance awareness and maturity* of the potential suppliers and → *request supplier certification or self-certification*.

Problem How can you use supplier contracts to address open source governance and compliance?

Solution Companies build their supplier relationships using supplier contracts with clear terms for the functionality, quality and availability of the supplied software. However, contracts do not address aspects of legal compliance specific to open source software components, which can cause potential governance and compliance problems, as the responsibility for the final software product that uses supplied code remains with the client and not the suppliers. Even if some contracts have clauses for putting partial responsibility for legal compliance caused by suppliers, this is not enough to ensure no risk caused by ungoverned use of open source in products. In case of a potential litigation, putting the blame solely on a supplier and potentially → *running a supplier out of business is not a good strategy*, as you will be left with no supplier, no source code in most cases and a remaining legal issue of license compliance in your product. Therefore, address this issue early on and prevent risks of open source compliance and governance by adding clauses on the issue in the supplier contract as early as possible. The contract should inform the supplier of all the obligations around open source license compliance and other aspects of FLOSS governance. Design contracts that outline a supplier's responsibility in case of license non-compliance, and outline preventive measures, such as → *performing certification or self-certification*, which can be optional or mandatory. Supplier contracts can also include stricter provisions, such as specific templates that a supplier must fill before any anticipated use of open source components in software development and send these templates to the client for approval.

You can then use the open source component information in the provided template to → *run through your component approval process* before allowing or rejecting the use of the component in the software that will eventually be supplied to you. Though more cost and labor intensive this approach makes sense for companies that have critical suppliers and large number of products that will have the supplied component. In such cases the potential risk of non-compliance is too high to only rely on contracts and subsequent → *supply chain audit*.

B.4 Corrective Governance

B.4.1 Audit your supply chain

Name	Choose the right supplier
Actor	Supply chain management responsibles, IT department
Context	You are → *undertaking preventive measures* to ensure open source governance and compliance. You already have many suppliers who supply critical software components that make it into your products.
Problem	How can you check and ensure open source governance and compliance in the supplied code beyond trusting the supplier contracts that you → *designed with FLOSS governance and compliance issues in mind*?
Solution	Even though supplier contracts specify that suppliers should ensure open source license compliance in the code they provide, some open source components go unnoticed by the suppliers and end up in your products, in which case you bear the responsibility for potential non-compliance or license and copyright violations. To prevent the resulting legal and financial risks, you should audit your supply chain by scanning and analyzing the supplied binaries and source code when possible. As a result of such audit checks you will identify governance and compliance issues in the supplied code, which can be communicated back to the suppliers with a request to fix the irregularities as specified in the supplier contract. Run two types of supply chain audits: • regular audits • surprise audits. For audits → *use tools* and document your findings in a database or in BOM documentation.

	If you identify a large number of compliance issues, consider → *changing a supplier*, but → *don't run the supplier out of business*.

B.4.2 Mitigate identified risks

B.4.3 Assess risks in accordance to the supply chain management policy

Name	Assess risks in accordance to the supply chain management policy
Actor	OSPO (Open Source Program Office), Supply chain management responsibles
Context	You are → *undertaking preventive measures* to ensure open source governance and compliance. You already have many suppliers who supply critical software components that make it into your products. You also → *audit your supply chain* to identify potential risks of open source compliance.
Problem	How can you assess potential risks of non-compliance caused by the supply chain?
Solution	After → *auditing your supply chain*, you should document the identified issues and risks in a systematic manner. You then should assess these risks in accordance to the → *established supply chain management policy*. To assess the risks develop a matrix of risk probability and criticality. Develop performance plans for different risk probability-criticality combinations. For risks that have a high probability and high criticality, the outcome of non-compliance will have the most negative impact on the company. So these risks should be prioritized and addressed first by → *triggering supplier contract clauses and get the supplier to take care of the issue*, but → *not run your supplier out of business*, as this would only increase the risk and the mitigation cost.

B.4.4 Trigger supplier contract clauses and get the supplier to take care of the issue

Name	Trigger supplier contract clauses and get the supplier to take care of the issue
Actor	OSPO (Open Source Program Office), Lawyer / legal counsel
Context	You have → *designed supplier contracts with open source governance in mind*. You have identified compliance and governance risks in your supply chain and → *assessed these risks in accordance to the supply chain management policy*.
Problem	What actions should you take to address the identified risks of non-compliance by a supplier?
Solution	For risks that have high probability and high criticality, the outcome of non-compliance will have most negative impact on the company. Such risks should be prioritized, and you should trigger supplier contract clauses and get the supplier to take care of the issue, but → *not run your supplier out of business*, as this would only increase the risk and the mitigation cost. When you → *designed supplier contracts with open source governance in mind*, you included the obligations of the suppliers. In case of non-compliance, these obligations are not met, thus the contract is violated, and the supplier is liable for the error. You should start by getting back to your supplier and communicating the issue. In most cases, the supplier will work on solving the issue without need for litigation. Otherwise, you can use legal tools to force compliance. In any case, it should be your last resort to sue a supplier or to shift responsibility to the supplier, as this can run the supplier out of business and leave you alone to deal with the issue.

B.4.5 Do not run your supplier out of business

Name	Don't 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 to 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 you 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 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.

B.4.6 BOM Management

B.4.7 Identify open source components and metadata from the supply chain

Name	Identify open source components and metadata from the supply chain
Actor	OSPO (Open Source Program Office), Supply chain management responsibles
Context	Managing your software supply chain you have conducted → *preventive governance measures* and → *corrective governance measures*, namely you have → *conducted audits of the supplied code*.
Problem	What should you about the open source components coming through supply chains?
Solution	Bill of materials management is a key aspect of open source governance in supply chains. Using bill of materials as a central artifact for FLOSS governance, companies can manage and map the supplied open source components they are using in their products. Bill of materials for the supplied software should be used to identify the open source components and their metadata. Together with the data from component approval and component reuse, this data will then be used to → *track, document and update BOM in a consistent and complete manner*, while → *using a machine-readable and standard format for BOM upon software supply*.

B.4.8 Track, document and update BOM in a consistent and complete manner

Name	Track, document and update BOM in a consistent and complete manner
Actor	OSPO (Open Source Program Office), Supply chain management responsibles
Context	You have used the bill of materials and code scanning of the supplied code to → *identify open source components and metadata from the supply chain*.

Problem	What should you about the identified open source components?
Solution	Use the data and metadata of the open source components from your component approval, component reuse and supplier management processes to track, document and update bill of materials for all your products in a consistent and complete manner. This will allow to generate an up-to-date view picture of all the open source components used on demand, thus enabling license compliant product release without delays or additional license checks common in the industry. You can use BOM to visualize the product architecture model and to integrate the open source component data with → *established component repository*, resulting in further synergy. You need to identify, track and document the open source components used by your suppliers and their suppliers to ensure the visibility of all supplier tiers and their open source use. Consider adding a clause in the → *designed supplier contracts* to require the provision of detailed BOMs with open source specific metadata, and → *use machine-readable and standard format for BOM upon software supply* to enable the easy use of tools for supplier management.

B.4.9 Have a backup of open source components hosted by yourself

Name	Have a backup of open source components hosted by yourself
Actor	OSPO (Open Source Program Office)
Context	You have used the bill of materials and code scanning of the supplied code to → *identify open source components and metadata from the supply chain*. You have → *tracked, documented and updated BOM in a consistent and complete manner*.
Problem	You are responsible for the open source components used in your products and for their compliance. How can you ensure that open source components are available to you and to your customers in the long term?

Solution	Open source components are often hosted by their developer communities or by foundations that are supporting certain open source projects. In the long term some of these communities and foundations cease to exist, which can result in interrupted hosting of the source code that you rely on. To avoid such issues, host a backup of the open source components that are part of your BOM by yourself. You can use this backup hosting for your → *component repository*. You can also use this hosting to share your own source code, if required by an open source license.

B.4.10 Use machine-readable and standard format for BOM upon software supply

Name	Use machine-readable and standard format for BOM upon software supply
Actor	OSPO (Open Source Program Office), Supply chain management responsibles
Context	You have used the bill of materials and code scanning of the supplied code to → *identify open source components and metadata from the supply chain*. You have → *tracked, documented and updated BOM in a consistent and complete manner*.
Problem	What can you improve the performance of managing your BOMs?
Solution	Software supply chains are complex and cannot be handled manually. You need to → use tools to improve the performance of BOM management. Most importantly you need to establish a machine-readable and standard format for BOMs. An example of such a format is called Software Package Data Exchange (SPDX). It enables the documentation and exchange of data and metadata for open source components and BOMs made of such components.

B.4.11 License Compliance for Supply Chain

B.4.12 Review identified open source components and metadata for license compliance

Name	Review identified open source components and metadata for license compliance
Actor	OSPO (Open Source Program Office), Supply chain management responsibles
Context	You have used the bill of materials and code scanning of the supplied code to → *identify open source components and metadata from the supply chain*.
Problem	How can you ensure license compliance for the identified components from the supply chain?
Solution	Once open source components are identified using BOMs, you need to check them license compliance by reviewing the components and their metadata. You go through the components checking their licenses and their use cases. If your company's policy allows the use of an identified license/use case pair, you should document it as compliant. If you find a non-compliant component, you → *need to mitigate the risks*. You can → *trigger a clause in the supplier contract to get a supplier to fix the non-compliant situation*. Alternatively you might have to change your supplier or replace the non-compliant component on your own.

B.4.13 Review license obligations in the context of supply chain management

Name	Review license obligations in the context of supply chain management
Actor	OSPO (Open Source Program Office), Supply chain management responsibles

Context	You have used the bill of materials and code scanning of the supplied code to → *identify open source components and metadata from the supply chain*. You have → *reviewed identified open source components and metadata for license compliance*.
Problem	What do you need to do to comply with open source licenses of the components from the supply chain?
Solution	You need to review the obligations of the identified open source licenses in order to comply with them. Different licenses have different obligations, which all have to be documented and checked with the use cases in which the components are used. Multiple licenses in one component need to be considered, as they can result in incompatible license mixtures. Use the reviewed license obligations to → *ensure license compliance*.

C

Data Gathering – Interview Questions

Appendix C presents the interview questions we developed for qualitative data gathering. Section C.1 presents the set of interview questions we created to gather data from the FLOSS governance experts for theory building. Section C.2 presents the set of interview questions we created to gather data from the case study companies' employees for initial situation assessment at case study companies during theory evaluation. We developed the interview questions in accordance with the employed research methods for theory building [77] and for theory evaluation [157].

C.1 Interview Questions – Expert Interviews for Theory Building

Section C.1 presents the interview questions we created for the expert interviewees from the sample of 15 companies with an advanced understanding of open source governance.

Interviewee Context

- Could you briefly present your company? What are your main products and respective markets?
- Could you briefly present yourself? What's your role at your company? What are your responsibilities?
- How are you involved with Open Source Software (FLOSS) at your job? Especially when using FLOSS in the products you sell / offer?

General FLOSS Governance Best Practices

- Do you have a strategy (strategic program) for Open Source Governance (especially dealing with FLOSS in products)? Could you please describe it?
- How do you use FLOSS in your products? Some examples?
  - How do you search for FLOSS components that you will use in your products?
  - How do you choose a FLOSS community to take code from? Do you have well-defined or ad-hoc criteria (age, diversity, activity of the community, etc.)?
  - How do you integrate various FLOSS components into your products? Do you use any interfaces to isolate some FLOSS components from your main code-base?
  - How do developers decide whether to use a given FLOSS code? What's the process and who is involved (checking license compatibility, legal review, approval process, etc.)?
  - How do you mark the FLOSS components used in your products and where they are used (software architecture)? How is this useful?

  - Do you have a centralized repository / knowledge sharing system to document FLOSS usage in your products / projects to ensure its reusability by other developers at your company?

- Do you contribute back to FLOSS communities you took code from?
  - How do you contribute? Some examples (bug fixes, functional improvements, issues raised, etc.)?
  - Why do you contribute? What are the tangible benefits to your products / projects or the company in general?
  - Do you have rules for your contributions?
  - Are you part / leader of a FLOSS foundation / community? How is does this help your company create better products?
  - What are some benefits or challenges for your engagement in FLOSS communities?
  - How much do you contribute to the FLOSS community? How do you decide when and where to contribute?
  - When do you not contribute to given FLOSS communities?
  - Are there critical communities for you and your products?
- Do you have product management-oriented tools to help deal with FLOSS governance and license compliance? Examples?
- Do you have development tools to help deal with FLOSS governance and license compliance? Examples?
- Are your products based mainly on FLOSS components or do you mainly use proprietary components?
- How did your FLOSS governance practice evolve over time?
- How did you recognize the need for Open Source Governance?

General FLOSS Compliance Best Practices

- Do you have a strategy (strategic program) for Open Source Compliance (especially dealing with FLOSS in products)? Could you please describe it?
- Do you have detailed processes of FLOSS license compliance? Some examples?
- Which licenses do you prefer when choosing FLOSS components to use in your products?
  - Do you give preference to standard (unmodified) licenses?
  - Which specific licenses (MIT license, GPL, etc.) do you use? Which ones are more common and why?
  - Are there some licenses whose use is especially problematic? Why?
  - Do you have a legal department with expertise in FLOSS compliance? How is this (could this be) beneficial?
    * Does legal department review FLOSS licenses before the code is integrated into your products? What's the process like?
    * Does legal department provide guidelines to the developers on which FLOSS licenses are OK to use and which ones need explicit legal review and approval? What form do these guidelines take?
    * Are the developers aware of various FLOSS licenses? Do they get relevant training?
  - How do you deal with situations when you need to mix various licenses in one product?
- Did you or do you scan your products' code to ensure FLOSS license compliance? Who and how does this scanning?
  - Which tools do you use for such scanning?
  - How often do you scan your code?
  - What are the benefits and challenges of scanning your code for FLOSS compliance?
- How do you deal with identified FLOSS license non-compliance? Any examples?

- How did your FLOSS license compliance practice evolve over time?
- How did they evolve over time? Any exceptional cases (company acquisition, lawsuits, etc.)?
- What was your first step in addressing limited FLOSS governance and non-compliance?

People Related Best Practices

- Who makes final decisions on the inclusion of certain FLOSS components into products?
- Do you have a board / team responsible for Open Source governance and compliance?
  - How is it structured? What are the roles of the members?
  - How does this team interact with other parts of your company?
  - How independent are this team / group from the rest of the organization? How do you collaborate internally, especially in terms of software engineering?
  - Is top management supportive of this board / team? Why?
- Do you have a compliance manager or a person specifically responsible for Open Source governance and compliance?
- How is your legal department involved in FLOSS licensing, license interpretation? Do they focus on compliance?
- Does legal department have a veto right in case non-compliance is detected?
- How is IT department involved in FLOSS governance and compliance processes?
- Who else is involved in FLOSS governance and compliance processes?

Supplier Management Related Best Practices

- Who are your main code suppliers? What the proportions of FLOSS vs. proprietary suppliers?
- Do you require proof of FLOSS compliance from them?
  - Do you use SPDX standard for this? Why?

  - What are the benefits and challenges of asking for bills of materials of supplied software?
  - Do you require any certificates from them (OpenChain, etc.)?
- How did your supplier management change over time? Did you encounter any challenges?
- Do you verify FLOSS compliance of supplied code?
  - How (scanning tools, manual review)? Is this automated? Do you use code scanners?
  - Do you require bill of materials including licenses used in the supplied code?
- Do you, in turn, provide a bill of materials for your customers? Are they asking for this?
- Do your customers care about FLOSS governance and compliance practices you use?
- How do you deal with export restrictions caused by FLOSS code? Did you have any challenges with this?

Other Questions

- How are you improving your FLOSS governance and compliance?
- Are you learning from the experience of the other companies concerning FLOSS governance and compliance?
- Do you have anything to add?

C.2 Interview Questions – Situation Assessment for Theory Evaluation

Section C.2 presents the interview questions we created for the initial situation assessment of corporate open source governance at the case study companies. The questions are split into topical subsets. Subsection C.2.1 presents the general questions for all the interviewed employees. Section C.2.2 presents the interview questions only asked the management employees if the related topics have not already been addressed in answers to the general questions. Section C.2.3 presents the interview questions only asked the engineering employees if the related topics have not already been addressed in answers to the general questions.

C.2.1 General Questions for All Employees

Interviewee Context

- What's your role at Company X / subsidiary? What are your responsibilities?
- Where is your department / group situated in Company X/subsidiary's organizational structure?
  - Are there departments that are more advanced in terms of open source governance?
- What products / software products are you working on?
- How independent are your team / group from the rest of the organization? How do you collaborate internally, especially in terms of software engineering?

Open Source Software (OSS) Usage Situation

- Do you have any experience of work with OSS or OSS components at Company X/subsidiary?
  - OSS in your products
  - OSS as development tools
  - OSS in R&D
- What do you know about OSS, its benefits or challenges?
- Does your management encourage or discourage the use of OSS in products?
  - Direct management
  - Top management
- Who makes decisions concerning OSS usage? Escalation? Final decisions? Is there an arbitration committee making decisions?
  - Buy vs make vs OSS decision making
  - Economic evaluation

- Is Company X/subsidiary or individual departments actively involved in an open source community? Would it be beneficial?
- Would you like to be active members of open source community? What are the perceived benefits and challenges?

Open Source License Compliance Situation

- How do you understand open source compliance?
- What OSS licenses do you use (LGPL, MIT License, etc.) and why? How much attention is given to licenses?
- Can you make decisions about open source compliance or OSS licenses in general?
- Who is responsible for OSS compliance in your team / department?
- Are you aware of risks of OSS non-compliance?
- Do you have a mechanism for assuring OSS license compliance?
  - Are they automated? Which tools are used?
- Do you think there is a need for OSS compliance process?
- Are internal or external customers asking
  - better OSS licensing information
  - bill-of-materials including OSS components
  - or license compliance?
- How is your legal department involved in OSS licensing, license interpretation? Do they focus on compliance?
- Does legal department have a veto right in case non-compliance is detected?

Open Source Governance Situation

- How do you understand OSS governance? What does it include?

- Do you have an OSS governance program or individual processes?
- Do you have rules / guidelines concerning OSS use? Do you follow them? Are they enforced?
- Did you read the documentation about OSS use at Company X/subsidiary? Where and what?
- Who makes decisions about OSS components, their usage, their quality assurance, licenses?
- How would you evaluate your OSS governance situation in your team / department?
- Is there a centralized body / team / responsible person to address for OSS related decisions?
- Did you have a need for OSS governance program / process / guidelines?
- Would you benefit from OSS governance program / process / guidelines? How, specifically?
- Do you identify potential risks of ungoverned OSS use or non-compliance? Do you match OSS governance policies to these risks?
- Would you want to see a centralized OSS governance and decision making at Company X/subsidiary? Why?
- Would you prefer to have an OSS expert in your team? Why?
- How is OSS governance and compliance handled in case of mergers and acquisitions?
- How would you report non-compliance or poor governance decisions?
    - Are there outlets for whistle-blowing?

C.2.2 Management Employee Questions

Open Source Software Usage Situation

- Do you use OSS in your products? Which (types of) products especially? Examples, reasons?
- Have you assessed (code scanning, surveys) the use of OSS in your products? Do you have an overview? How does it affect your products?

- Do you have a product architecture model including OSS components?
- Do you track your usage of OSS components and licenses? How? With which tools?
- Did you scan your code to find which OSS components are in use and their compliance? How do you control and manage OSS components in your products?
- Do you think there is a need for more OSS use in your products?
- Which products make better use of OSS? Why?
- Do you have regular training / internal education on OSS usage or governance for product managers?

Open Source License Compliance Situation

- Do you analyze license compliance of your product as a whole?
- Do you realize a risk exposure analysis potentially caused by license non-compliance?
- How do you mitigate intellectual property risks, especially those related to patents?
- Are there pre-approved for use open source licenses?
- Do you have agreed-upon license interpretations?
- Do you develop standard license interpretation for products together with your legal team?
- Do you use (did you develop) standard license compatibility matrix to manage licenses in your products? Do you enforce it on engineers?
- Do you define OSS component requirements for your developers? How detailed?
- Do you have OSS governance requirements for your suppliers?
  - Do you require a bill-of-materials (standards) with OSS components (and their licenses) used in the supplied software?
  - Do you automate this? Why?
  - Are you using a standard format? Why?
- Do you audit the supplied code / products to check compliance?

Open Source Governance Situation

- Do you have an OSS governance program for product managers? Any rules, guidelines documented?
- Do you have an OSS governance role / team? Who has what role?
- Do you do a capabilities analysis to assess your need and dependence on OSS?
- Have you defined product management goals of OSS governance?
- How do you communicate these goals to your team and across teams? Who's responsible?
- How important is good OSS governance for your clients? Why? How does it show?
- Who do your customers contact to discuss licensing and compliance of OSS components of your products?
- Do you keep an eye on industry best practices and standards for managing OSS use in products? Are you involved in a community / network?
- Do your suppliers use OSS? Do you track this usage and compliance?
- Do you ask for better OSS licensing or compliance / governance certification from your suppliers?

C.2.3 Engineering Employee Questions

Open Source Software Usage Situation

- Do you use OSS in development? How much? Some examples?
  - For development (i.e. developer tools e.g. Eclipse Java IDE)
  - In products (e.g. code components, libraries, e.g. glibc, Linux, Apache commons library)
- What OSS components are mainly used by you (your team)?

- Which OSS components are especially important / crucial for the development of your products?
- Do you modify and contribute to open source communities? Why? How?
- Are used OSS components shared throughout Company X/subsidiary or between teams? Do you reuse them?
- Do you have a repository for used OSS components? Would you need one? Why?
- Does this component repository have good search mechanism in accordance with requirements by product management?
- Describe your software development practices, especially concerning OSS selection and usage. Preferably on a specific example.
- Do you track your usage of OSS components and licenses? How? With which tools?
- Did you scan your code to find which OSS components are in use and their compliance? How?
- Do you manage / analyze OSS contributions to your code? How (survey, external tools)?
- Did you consider using a mandatory survey for initial assessment of OSS contributions?

Open Source License Compliance Situation

- Do you have license-related rules about OSS components used in development? What licenses can you use, what licenses are not allowed?
- What OSS licenses are mainly used by your team or at Company X/subsidiary in general? Which licenses are given priority (LPGL, MIT License, etc.)? Examples, if possible.
- Are you restricted in the source code that you can use in development, based on the original license?
- When using OSS components, do you ask for license vetting?
- Did you have issues of OSS non-compliance? How do you detect these issues?

- What do you do if non-compliance is detected? Do you have a process? Is it case-based?
- Do you replace problematic components? How? Practice examples?
- Do you decouple problematic components? How? Practice examples?

Open Source Governance Situation

- Do you have an OSS governance program for engineers? Any rules, guidelines documented?
- Do you have quality-related concerns when using OSS components? Why? How do you address any?
- Do you have a process for quality assurance of OSS code? Please describe and give an example if possible.
- Do you analyze security risks of OSS involvement?
- Did you have regular training / internal education on OSS usage or governance for engineers, especially new developers?
- Who approves selection and use of OSS components?
- If there an approval process for OSS components that a developer wants to use? Are these decisions centrally stored and available to all developers?
- What would you like to see as part of OSS governance handbook?

C.3 Interview Questions – Theory Evaluation at Case Study Companies after Handbook Implementation

Section C.3 presents the interview questions we created for the theory evaluation at the case study companies, where we guided the implementation of our handbook for corporate open source governance – an actionable presentation of our theory. The questions are split into subsets according to the units of analysis we defined for the case studies following Yin's methodology [157] and according to our Case Study Protocol presented in Appendix E. Subsection C.3.1 presents the general questions for all the interviewed employees. Section C.3.2 presents the interview questions on the general

aspects of the handbook implementation, its preparation, and execution at the studied companies. Section C.3.3 presents the specific questions with the predefined theory evaluation criteria asked for three units of analysis: handbook as a whole, handbook sections, and individual best practices. For the latter, we focus on select best practices from our theory for an in-depth evaluation.

C.3.1 Questions on Interviewee Context

Interviewee Context

- What's your name and role at your company?
  - What are your responsibilities?
  - Which organizational unit are you part of?
- How are you involved with open source governance at your company?
- How are you involved with open source governance handbook implementation at your company?

C.3.2 Questions on General Aspects of Handbook Implementation

General Aspects of Handbook Implementation

- Before implementation, how did you assess the current situation of corporate open source governance at your company?
- How did you prepare for handbook implementation?
  - Who were the involved stakeholders? What tasks do they have?
  - Please describe the planning process?
  - Which part of the handbook did you start with? What came next? Why?
- How did you conduct the actual implementation?
  - Who were the involved stakeholders? What tasks do they have?

  - Please describe the actual implementation process?

- What were the main artifacts used in handbook implementation?
- What were the main tools used in handbook implementation?
- What were the main challenges and issue during implementation?

C.3.3 Questions on Theory Evaluation via Handbook Implementation

Level of Evaluation - Handbook as a Whole

- How *complete* was it? Did it have an adequate beginning, middle, and end? Did it lack anything?
- How *variable* was it? Did it have a mixture of concepts, not focusing on single concepts?
- How *well-structured* was it? Are the parts structured in a logical and interconnected manner?
- How *comprehensive* was it? Did it answer all the problems you had? Did it go into enough detail?
- How *understandable* was it? Did you and other employees understand the intention and the specifics?
- How *applicable* was it? Were you able to apply it in your context? Did you need to adjust anything?
- How *relevant* was it? Did it address an issue of relevance for the company and employees?
- How *significant* was it? Was the impact on the company significant?
- How *useful* was it? Did it add value to your company in solving the issue? Did it enhance your knowledge on the issue? Did it achieve its goals?

Level of Evaluation - Section - Getting Started

- *[Question X]* In the context of applying this handbook section:

  - How *complete* was it? Did it have an adequate beginning, middle, and end? Did it lack anything?
  - How *well-structured* was it? Are the parts structured in a logical and interconnected manner?
  - How *comprehensive* was it? Did it answer all the problems you had? Did it go into enough detail?
  - How *understandable* was it? Did you and other employees understand the intention and the specifics?
  - How *applicable* was it? Were you able to apply it in your context? Did you need to adjust anything?
  - How *relevant* was it? Did it address an issue of relevance for the company and employees?
  - How *significant* was it? Was the impact on the company significant?
  - How *useful* was it? Did it add value to your company in solving the issue? Did it enhance your knowledge on the issue? Did it achieve its goals?

Level of Evaluation - Best Practice - OSGOV-GETSTA-TRAORG-1. Establish a board of stakeholders to organize the transition (A.1.1)

- Which stakeholders were on the board of transition?
  - How did you choose them?
  - Why them?
- *[Question Y]* In the context of applying this best practice:
  - How *well-structured* was it? Are the parts structured in a logical and interconnected manner?
  - How *comprehensive* was it? Did it answer all the problems you had? Did it go into enough detail?

  - How *understandable* was it? Did you and other employees understand the intention and the specifics?
  - How *applicable* was it? Were you able to apply it in your context? Did you need to adjust anything?
  - How *relevant* was it? Did it address an issue of relevance for the company and employees?
  - How *significant* was it? Was the impact on the company significant?
  - How *useful* was it? Did it add value to your company in solving the issue? Did it enhance your knowledge on the issue? Did it achieve its goals?

Level of Evaluation - Best Practice - OSGOV-GETSTA-TRAORG-4. Start small, then replicate - define the scope of the transition process (A.1.4)

- What was the scope of the transition process?
- Did you follow the process matching the one from the handbook? Did you have to modify it? How?
- In the context of applying this best practice, the same questions as in *[Question Y]*.

Level of Evaluation - Best Practice - OSGOV-GETSTA-TRAORG-5. Define the transition timeline (A.1.5)

- What was the timeline of the transition process?
- Did you follow the process matching the one from the handbook? Did you have to modify it? How?
- In the context of applying this best practice, the same questions as in *[Question Y]*.

Level of Evaluation - Best Practice - OSGOV-GETSTA-IPRISK-1.3. Create license/use case pairs (A.4.4)

- How did you create license/use case pairs during the transition towards governance?

- How did you document the license/use case pairs?
- In the context of applying this best practice, the same questions as in *[Question Y]*.

Level of Evaluation - Section - General Governance

- Same as in *[Question X]*.

Level of Evaluation - Best Practice - OSGOV-GENGOV-GOVMAN-3. Establish open source program office

- Same as in *[Question Y]*.

Level of Evaluation - Best Practice - OSGOV-GENGOV-CAPABI-2. Create educational resources for capabilities building

- Same as in *[Question Y]*.

Level of Evaluation - Section - Inbound Governance

- Same as in *[Question X]*.

Level of Evaluation - Best Practice - OSGOV-INBGOV-COMAPP-1. Define the component approval process

- Same as in *[Question Y]*.

Level of Evaluation - Best Practice - OSGOV-INBGOV-COMREU-8. Create component repository

- Same as in *[Question Y]*.

Level of Evaluation - Section - Outbound Governance

- Same as in *[Question X]*.

Level of Evaluation - Best Practice - OSGOV-OUTGOV-LICCOM-1. Ensure license compliance

- Same as in *[Question Y]*.

Level of Evaluation - Best Practice - OSGOV-OUTGOV-CONMAN-1. Establish contribution management policy

- Same as in *[Question Y]*.

Level of Evaluation - Section - Supply Chain Management

- Same as in *[Question X]*.

Level of Evaluation - Best Practice - OSGOV-SUCHMA-SCMPRO-2. Establish supply chain management process

- Same as in *[Question Y]*.

Level of Evaluation - Best Practice - OSGOV-SUCHMA-SCMPRO-5. Use tools to automate supplier management

- Same as in *[Question Y]*.

Level of Evaluation - Best Practice - OSGOV-SUCHMA-BOMMAN-1. Identify open source components and metadata from the supply chain

- Same as in *[Question Y]*.

Level of Evaluation - Best Practice - OSGOV-SUCHMA-BOMMAN-2. Track, document and update BOM in a consistent and complete manner

- Same as in *[Question Y]*.

Level of Evaluation - Best Practice - OSGOV-SUCHMA-BOMMAN-4. Use machine readable and standard format for BOM upon software supply

- Same as in *[Question Y]*.
- Do you have any additional remarks about the implementation of the handbook, its content, or process?

D

Qualitative Data Analysis – Code Systems

Appendix D presents the code systems developed for and used in the qualitative data analysis in this research project. We used qualitative data analysis for the literature review presented in Chapter 2, and for theory building presented in Chapter 3. This appendix gives an overview of the code systems. See Section D.1 for the code system for the literature review, and Section D.2 for the code system for the theory building.

D.1 QDA Code System – Literature Review

Code System				Coded Segments	Documents
All Codes				1443	87
OS Adoption				0	0
	Categories			5	3
	Maturity And Measurement			19	6
	Decision Criteria			11	6
		OS And Standards		32	6
	Open Innovation			15	7
	OS Business Model			25	14
		Creating A Business Model		1	1
	OS Evolution			15	11
		Commercially Friendly Licensing		6	4
		Standards		23	10
	OS Challenges			31	12
		Leadership And Control		4	2
		Others		17	10
		Switching Costs		9	6
		Lower Innovation		4	4
		Free Riding Problem		7	6
		Company's Readiness		6	6
		Infringements Due To Licenses		10	9
		Support		4	3
		Development Issues		9	7
		User Motivation		8	6
		Edge Case Groups		3	3
	OS Benefits			24	15
		Standards		3	2
		Others		12	9
		Skills		5	5
		Security		2	2
		Customization		3	3
		Popularity		10	7
		Benefits For Public Sector		7	3
		Standards Adoption		1	1
		Trialability		3	3
		Reliability		7	5
		Development Enhancements		10	7
		Projects As Examples For Success		10	7
		Economic Efficiency		1	1
			Profit	8	8
			Quality	7	6
		Costs		15	10
		Competitive Advantage		7	5
		Innovation		6	5
	Current Adoption Of OS			8	7
Getting Started				0	0

Code System				Coded Segments	Documents
	Education And Communication			2	1
		Communication		1	1
		Education		3	3
	Risk Analysis			5	4
		Security Risk Analysis		2	2
		IP-at-Risk Analysis		4	2
	Contribution Analysis			3	2
	Product Architecture			3	3
	Product Analysis			7	5
	Transition To Governance			2	2
		Transition Motivation		2	2
		Transition Board		2	2
		Transition Policy		5	2
OS Progam Office				0	0
	Purpose			4	4
	Goals And Responsibilities			12	8
	Members And Structure			6	4
	Head Of Program Office			1	1
	OS Program			0	0
		Best Practices		1	1
		Artefacts		1	1
		Processes		1	1
		Principles		2	1
Governance Management				2	2
	Best Practices			15	6
	Purpose			9	6
	Conflict Resolution			1	1
	Decision Making			6	6
	Management Hierarchy			3	3
	Governance Processes			6	4
	Strategy And Policy			8	4
		Policy		16	7
		Strategy		11	6
		Challenges		10	5
		Principles		4	3
License Interpretation				2	2
	Best Practices			2	2
	Distribution of Licenses			9	4
	OS Definition			17	15
	License-Use Case Pairs			7	4
	Technical Interpretation			5	4
	Business Interpretation			15	5
		Edge Case Licenses		5	4
		Permissive Licenses		7	4
		Licenses In General		14	11
		Reciprocal Copyleft Licenses		18	12
			GPL	6	6
		Purpose And Definition		17	10
		Dual Licensing		1	1

Code System				Coded Segments	Documents
	Legal Interpretation			0	0
		Edge Case Licenses		12	8
		Reciprocal (Copyleft) Licenses		16	8
		Permissive Licenses		6	5
		General OS Licenses		9	8
Communication Education				0	0
	Organizational Capabilities			0	0
		Learning Materials		6	2
		Employee Training		13	11
		Assessing Employee Capabilities		2	2
	External Communication			2	2
		Scientists		1	1
		Public		10	6
		Suppliers		2	2
		Customers		2	2
	Internal Communication			1	1
		Knowledge Exchange		6	5
		Documentation		13	7
			Best Practices	6	3
			Challenges	7	2
		Communication Process		3	3
		Communication Channels		4	2
Incoming OS				0	0
	Software Development			8	7
		Best Practices		13	4
		Challenges		16	11
		Definition And Approaches		9	5
		Tools And Standards		5	3
		Other Development		2	2
		Component-Based Development		13	6
	Supplier Management			3	3
		Bill Of Materials Management		6	5
		License Management		3	2
		Planning		5	4
		Quality Management		4	3
		Certification		5	2
		Standards		10	6
		Audits		1	1
		Contracts		3	2
		Tools		8	1
		Optimization		9	5
		Challenges		8	7
	Component Monitoring			6	3
	Component Integration			6	5
		Best Practices And Workarounds		4	3
		Challenges		8	5
		Process		1	1
	Component Repository			9	6
		Tools		16	6

Code System				Coded Segments	Documents
		Product Architecture Model		4	3
		Reuse		11	7
		Search		8	3
		Documentation		7	3
	Component Approval			3	3
		Tools		3	2
		Approval Criteria		11	7
		Quality Assurance		2	2
		Process		4	4
	Component Selection			7	7
	Component Search			2	2
Outgoing OS				0	0
	License Compliance			2	2
		Legal Team		10	8
		Best Practices		25	10
			Examples Of Projects	2	2
			Questions To Be Asked	1	1
		Tools and Automation		15	9
		Patents		11	7
			Definition	1	1
			Challenges	3	2
		Mixing Licenses And Compatibility		10	8
		Challenges		26	10
		Obligations		12	5
		Compliance Process		19	6
			Release Review	4	3
		Compliance Officer Responsibilities		8	7
		Compliance Officer		4	4
Governance Research				0	0
	Research Issues			7	4
		Innovation		2	2
		Competing With Proprietary Software		2	2
		Licensing		2	1
		Choosing OS		3	3
		Development		2	2
		Contribution		5	4
	Literature Review			0	0
		Literature Review Results		6	2
		Literature Categories		3	2
		Definition Of OS Governance		7	3
OS User Foundation				5	5
OS Developer Foundation				6	5
OS Communities				28	11
Private Contribution				1	1
	Benefits			9	5
	Motivation			24	11
Corporate Contribution				2	2
	Decision Criteria			6	3
	Best Practices And Strategies			7	6

Code System		Coded Segments	Documents
Types Of Contribution		18	8
	Roles Within The Community	6	3
Challenges		25	10
Benefits		16	9
Motivation		36	20

Table D.1: QDA Code System – Literature Review

D.2 QDA Code System – Theory Building

Code System			Coded Segments	Documents
All Codes			1012	43
Getting Started			0	0
Transition Org			0	0
	Implement the transition process		25	10
	Communicate the transition process		14	7
	Establish the transition process		21	9
	Define the transition timeline		5	1
	Define the scope of the transition process		3	3
	Define responsibilities of the transition manager		6	3
	Designate the transition manager		9	7
	Establish a board of stakeholders		26	13
Transition Policy			0	0
	Adjust and improve FLOSS governance policy for the transition		3	3
	Communicate FLOSS governance policy for the transition period		2	2
	Establish FLOSS governance policy for the transition period		49	14
Product Analysis			0	0
	Use a combination of methods for product analysis		0	0
		Select and use governance tools for automation	24	13
		Establish a process of continuous reporting and assessment	17	8
		Use one mandatory survey for initial assessment	3	3
	Product architecture model		0	0
		Maintain product architecture model	10	7
		Create product architecture model	12	8

Code System			Coded Segments	Documents
	Method of product analysis		0	0
		Use one mandatory survey for initial assessment	1	1
		Establish a process of continuous reporting and assessment	4	4
	Contribution Analysis		0	0
		Create a policy on how to contribute back to the community	15	8
		Assess contributions by survey	1	1
		Assess contributions by external tools	1	1
	Use Analysis		0	0
		Document current open source use	18	11
		Run open source use analysis in products	5	3
IP-at-Risk Analysis			0	0
	License Compliance Analysis		0	0
		Create license-use case pairs	12	7
		Use standard license interpretation	27	13
		Develop standard license interpretation	29	14
	Risk Exposure Analysis		28	11
	IP Risk Mitigation		0	0
		Replace problematic components	5	4
		Decouple problematic components	4	2
		Require bill-of-materials	19	10
		Run random audits	14	9
	Security Risk Analysis		9	6
	Capabilities Analysis		2	2
Communication and Capabilities			0	0
	Establish communication channels for open source governance han		14	7
	Assess open source governance capabilities		2	1
	Develop FLOSS governance and compliance capabilities at the cen		10	1
	Design employee training		6	6
	Provide employee training		10	9
Governance			0	0
General Governance			0	0
	Governance Management		0	0
		Define goals of governance	9	7
		Establish an open source program	7	5
		Establish an open source program office	7	5
		Define role of legal counsel	5	4

Code System			Coded Segments	Documents
	Give legal counsel veto right		1	1
	Give arbitration committee decision right		1	1
	Integrate program office in product development		2	1
	Integrate program office in mergers and acquisitions		2	1
Open Source Program Office			0	0
	Define roles, responsibilities, and policies		21	12
	Provide roles, responsibilities, and policies in written form		5	4
	Match policies to actual risks		7	6
	Provide contact for internal inquiries		5	3
	Provide channel for whistleblowing		2	2
	Provide contact for external inquiries		2	2
	Collaborate with legal counsel on license interpretation		3	3
	Track industry best practices and standards		9	5
	Network to learn from others		8	4
	Engage with community		5	3
License Interpretation			0	0
	Use standard license compatibility matrix		8	7
	Develop standard license compatibility matrix		6	5
	Automate license identification and interpretation		10	7
Capabilities			0	0
	Capabilities Analysis		0	0
		Assess open source governance capabilities	1	1
	Capabilities Building		0	0
		Create educational resources for capabilities building	4	2
Inbound Governance			0	0
Component Selection			8	6
	Selection criteria		3	1
	Component selection process		3	3
Component Search			3	3
Component Approval			7	6
	Review a component approval request		0	0
		Review use in context of product architecture	3	2

Code System			Coded Segments	Documents
		Analyze code for license compliance	6	4
	Approval process		0	0
		File a component approval request	5	2
		Define component approval process	5	5
		Communicate component approval process	3	3
		Implement component approval process	4	4
		Appeal a component approval decision	1	1
		Make a component approval decision	2	2
		Communicate open source component approval rules	2	2
		Define transparent rules for open source component approval	4	3
	Documentation		1	1
		Add decision to component repository	7	4
		Provide approval request templates	3	2
Component Repository			6	3
	Component reuse policy		0	0
		Establish component reuse policy	7	7
		Communicate component reuse policy	3	3
		Adjust and improve component reuse policy	1	1
	Component reuse process		0	0
		Designate a role of responsibility for the component repository	1	1
		Establish component reuse process	1	1
		Communicate component reuse process	2	2
		Implement component reuse process	2	2
	Component repository		0	0
		Create a component repository	8	4
		Update component repository	9	7
		Maintain component repository	5	4
		Audit component repository	5	4

Code System			Coded Segments	Documents
		Use tools to create, update and maintain component repository	10	7
		Provide component repository a single well-defined location	3	2
	Track prior approval data for reuse		1	1
		Previous use information	5	5
		Component metadata	7	5
		Component data	3	3
	Provide all relevant meta-data for component		3	2
	Search component repository for reusable components		2	1
	Contact OSPO for details on a repository entry		2	2
	Add security check information to repository		2	2
	Link BOM and component repository		2	2
Supply Chain Management (SCM)			0	0
	Meta		0	0
		Challanges	6	4
		Opportunities	1	1
	Process		0	0
		Designate role responsible for supply chain management	4	3
		Establish supply chain management process	3	2
		Implement supply chain management process	1	1
	Policy		0	0
		Establish supply chain management policy	5	5
		Define tasks for the designated role responsible for supply chain	1	1
		Communicate supply chain management policy	3	2
		Adjust supply chain management policy	1	1
	Preventive Governance		0	0
		Choose the right supplier	0	0
		Assess open source governance and compliance awareness	3	3
		Assess governance maturity	3	3
		Run supplier self-certification	3	3
		Run third party supplier certification	1	1

Code System			Coded Segments	Documents
		Design supplier contracts with FLOSS governance aspects in mind	2	2
		Use contracts for open source tools for development (IT procure	2	2
		Use contracts for open source software for production - third p	3	2
	Corrective Governance		0	0
		Audit your supply chain	0	0
		Regular audits on supplier site	2	2
		Regular audits on own site	3	3
		Surprise audits	2	2
		Mitigate identified risks	2	2
		Assess risks in accordance to the SCM policy	4	4
		Trigger supplier contract clauses and get the supplier to take	2	2
		Don't run your supplier out of business	2	2
		Get the source code (before changing the supplier)	2	2
	(Special Aspect) BoM Management		0	0
		Identify open source components and metadata from the supply ch	3	3
		Document BoM in a consistent and complete manner	9	8
		Track open source component from the supply chain	5	5
		Keep BoM up-to-date	8	6
		Have a backup of open source components hosted by yourself	1	1
		Manage open source software on deeper levels in supply chain	3	2
		Use machine readable BoM upon software supply	17	10
		Use standard format for BoM upon software supply	17	10
	(Special Aspect) License Compliance for Supply Chain		0	0
		Review license obligations in the context of SCM	4	3
		Review identified open source components and metadata	12	5
		Review license obligations	6	5

Code System			Coded Segments	Documents
		Review copyright notices in the context of SCM	1	1
		Review security vulnerabilities in the context of SCM	5	4
	(Special Aspect) Tooling / Automation		0	0
		Tools for preventive governance	9	6
		Tools for corrective governance	1	1
		Tools for BoM management	5	5
		Integration with component approval and component repository (i	3	3
		Integration with license compliance and release management	3	2
Component Integration			4	2
Component Monitoring			3	2
Engineering Management			8	5
Communication			3	3
Education			2	2
	Exchange best practices and learn from others		6	3
			0	0
Release Management			6	5
	Concudct Source Code Inspections before Release		3	3
	Define Product Shipment Checklist		1	1
	Double check the contribution		5	4
Ensure License Compliance			18	10
	Disclose All Licenses Used		5	4
	Establish Compliance Policy		4	4
	Define Required License Artifacts		3	3
	Distribution Preparation		7	7
Contribution Management			18	8

Table D.2: QDA Code System – Theory Building

E

Evaluation Case Study Protocol

Appendix E presents the case study protocol for the multiple-case case study used during theory evaluation. The evaluation and its methodology are presented in Chapter 4. This case study protocol is developed following the case study research method by Yin [157].

E.1 Protocol Summary

We conducted expert interviews to develop a theory of industry best practices for open source governance in companies that use open source components in their products. We case our theory as a handbook of industry best practices. We plan to evaluate our theory by implementing subsets of the handbook best practices at 2-3 case study companies, where employees will follow these practices in production projects. We will measure the completeness, comprehension, understandability, applicability, and usefulness of the practices in order to evaluate the quality of our proposed theory. This document serves as a case study protocol and lays out the design of the case study. It outlines and discusses the case study design and planning we set out in preparation for the theory evaluation.

E.2 Case Study Overview

E.2.1 Case Study Research Questions

To gather data during the case study at each case, we plan to create two sets of questions – one for the initial situation assessment of open source governance at companies, another one for the theory evaluation at case study companies after the open source governance handbook implementation. For both questionnaires we plan to ask the following types of questions (building upon Yin's recommendations for case study protocol questions [157]):

- Level 1: questions on specific interviewees and their context
- Level 2: questions about case study companies (individual cases)
- Level 3: questions about patterns of findings across multiple case studies
- Level 4: questions about the entire case study
- Level 5: questions about policy recommendations and conclusions

The overarching goal of our research is to evaluate the quality of our theory on industry best practices for open source governance in companies. The quality criteria (based on related work in other disciplines [133] [11] [13] [85]) we will study are:

- *Completeness*
- *Variability*
- *Structure*
- *Comprehension*
- *Understandability*
- *Applicability*
- *Relevance*
- *Significance*
- *Usefulness.*

We will observe the subjects of implementation and conduct follow-up interviews to evaluate the quality criteria in relation to our proposed theory's implementation.

The research questions we will ask are:

- *RQ1*: How complete was it? Did it have an adequate beginning, middle, and end? Did it lack anything?
- *RQ2*: How variable was it? Did it have a mixture of concepts, not focusing on single concepts?
- *RQ3*: How well-structured was it? Are the parts structured in a logical and interconnected manner?
- *RQ4*: How comprehensive was it? Did it answer all the problems you had? Did it go into enough detail?

- *RQ5*: How understandable was it? Did you and other employees understand the intention and the specifics?
- *RQ6*: How applicable was it? Were you able to apply it in your context? Did you need to adjust anything?
- *RQ7*: How relevant was it? Did it address an issue of relevance for the company and employees?
- *RQ8*: How significant was it? Was the impact on the company significant?
- *RQ9*: How useful was it? Did it add value to your company in solving the issue? Did it enhance your knowledge on the issue? Did it achieve its goals?

E.2.2 Case Selection

For a case study, we searched for an organization that fulfills the following requirements:

1. Develops software and uses it in own (shipped) products
2. Uses open source software as part of own (shipped) products
3. No or basic open source governance in place.

We selected the following companies (anonymized per their request): Company A, Company B, and Company C. These are appropriate company choices from our sample because they meet our requirements, and have different levels of open source governance maturity, which enables a broader evaluation of the proposed theory.

E.2.3 Theoretical Framework

The theoretical framework for the case study is our theory of corporate open source governance, presented in detail in Chapter 3. Our proposed theory covers the key aspects of

corporate open source governance, including getting started with governance, general governance, inbound governance, outbound governance, and supply chain management governance.

In this theory evaluation case study, we will use our handbook for open source governance as a practically applicable representation of our theory. The handbook, whose parts are presented in Appendix A and in Appendix B, provides best practices for establishing and managing open source software governance and compliance at a company. It starts with the assumption that the company has decided that denying itself the benefits of open source software is counterproductive and that it would like to engage. This handbook provides an answer in the form of interlinked best practice descriptions. A best practice is presented as a pattern, that is, in the form of a context, problem, solution triple as known from the patterns community. All best practices thereby show how, why, where and by who they are applicable and what might come next. In this way, the handbook is more than a passive list of best practices; it provides active guidance for establishing and managing open source software governance and compliance at a company.

E.2.4 Case Study Design

There are different kinds of case studies. Case studies can differ on the purpose of research, whether different units of analysis are investigated, and whether they study the phenomena in different contexts or not. The next section will discuss why we perform an exploratory and descriptive (purpose), holistic (one primary unit of analysis), multiple-case study (multiple contexts).

E.2.5 Case Study Purpose

According to Yin [157] and Runeson [132] case studies can be used for different purposes:

- Descriptive – portraying the current status of a situation or phenomenon.
- Exploratory – finding out what is happening, seeking new insights, and generating ideas and hypotheses for new research.
- Explanatory – seeking an explanation for a situation or a problem, mostly but not necessarily, in the form of a causal relationship.
- Improving – trying to improve a certain aspect of the studied phenomenon - this is very close to action research [157]

To address our case study's overall goal of testing (evaluating) the proposed theory of open source governance, we set out two specific purposes for this study:

- Descriptive – in the first stage of the case studies, describing the situation of open source governance or lack thereof in detail.
- Exploratory – in the first stage of the case studies, explaining the handbook implementation at companies, its effects, and its quality criteria.

E.2.6 Unit of Analysis

According to Yin [157], there are two different units of analysis in case studies:

- Holistic – the case is studied as a whole.
- Embedded – the case is divided in multiple units (each being an individual data source during the case study).

Our case study will have one unit of analysis – the core teams at each case implementing and using our handbook for open source governance. These teams will be our main data sources. We will analyze how these teams are using the handbook and by extension how

our theory worked in their contexts (analyzing transferability of our proposed theory). We will break down our theory according to the logical categories (topic) of open source governance corresponding to the handbook sections, which will be implemented at different case study companies (depending on their open source governance maturity).

E.2.7 Case Study Contexts

Yin [157] proposes two types of case studies according to the study contexts:

- Single case case studies – analyzing one case only focused on one context
- Multiple case case studies – analyzing and comparing multiple cases with different contexts.

We will conduct a multiple case case studies with 2-3 case study companies in order to test our proposed theory in different context (in different production level environments). This will help us evaluate our theory from different perspectives and identify limitations to its transferability caused by real life application in different contexts.

E.2.8 Quality of Research Design

Following Yin [157], throughout the study we will ensure the quality of our case study research design by assessing its construct validity (by following case study research methods to identify correct operational measures for the evaluated concepts, and by ensuring data triangulation using multiple sources of evidence), internal validity (establishing causal relationships during handbook evaluation in the second explanatory stage of the study; not testable for the first descriptive stage of the study [157]), external validity, and reliability (by documenting out case study protocol to enable the replication of our study).

E.3 Data Collection Procedures

Yin [157] lists six potential sources of evidence:

- Documentation
- Archival records
- Interviews
- Direct observations
- Participants observation
- Physical artifacts.

We will aim to use as many data collection techniques and sources as possible. However, we do expect that our access to the internal documentation and archival records could be limited, given that we will working at production level projects at case study companies. Nor do we expect to deal with physical artifacts.

We will employ direct observation and participants observation throughout the case study, writing meeting minutes and notes for further analysis and case study results reporting. In particular we will directly observe the handbook implementation at the pilot projects and engaged teams at each case study company.

As our main source of data, we will use semi-structured interviews, interviewing employees in different roles related to open source use and governance. See Section E.4 of this protocol for details on the interview questions. Some of the potential weaknesses of this technique can be minimized in the following ways:

- Response bias – we will ask many individuals the same questions

- Inaccuracies and poorly articulated questions – we will develop interview questions based on our proposed theory of FLOSS governance, peer review the questions and improve them over time
- Reflexivity – we will only ask questions which do not imply answers.

In our case study we will interview employees from Company A, Company B, and Company C. If some interviewees can grant us access to documentation we will use the documentation for our case study, too.

Data collection and analysis will be conducted by the principal researcher (the author of this dissertation), and by colleagues and student helpers working temporarily with the principal researcher.

The interview records will only be available to the researchers of our research group at FAU (hosted on the university network). We will transcribe the interviews using services that are approved by case study companies in accordance to their data handling and privacy guidelines. We will get non-disclosure agreement from the transcription service companies used. The interview transcription will only be available to the researchers involved directly in this study. Both the transcriptions and the audio files will be archived in the case study database.

E.4 Data Collection Questions

To collect data, among other techniques, we will conduct interviews with the case study company interviews for our theory evaluation. We plan to collect data at two stages in each case study company:

- *Stage 1*: before the introduction of our theory at case study companies – questions on the initial situation assessment of corporate open source governance

- *Stage 2*: after the introduction of our theory at case study companies – questions on the handbook implementation and evaluation.

We will develop two detailed questionnaires, and improve them iteratively (addressing unclear questions or other issues we observe during the interviews). See the final questionnaire for *Stage 1* in Section C.2 in Appendix C. See the final questionnaire for *Stage 2* in Section C.3 in Appendix C.

For each stage we will select the most competent employees with the help of our contacts at case study companies. We will make sure that we interview employees with different and complementing roles to ensure the breadth of our findings.

E.5 Case Study Reporting

After collecting data from the case studies, we will analyze the data focusing on:

- the initial situation of open source governance
- the implementation of the governance handbook (select sections and best practices).

First, we will analyze the strengths, weaknesses, opportunities, and threats of the use of open source at each case study company. We will then analyze the initial (before the introduction of our governance handbook) governance situation at each case study company. We will report these situations in comparable formats across the case studies.

Second, we will analyze the implementation of the select sections of the corporate open source governance handbook, as well as on the select best practices from these sections. Discussing the handbook sections and individual best practices that were applied and used at case study companies, we will discuss how they improved the state of governance at each case study company compared to the respective initial governance situations. In reporting

our case study findings, we will have subsections corresponding to the evaluated parts of our theory in each case study report talking about the implementation, created company-internal artifacts, and proposed industry best practices (from our theory) used at each company. We will share handbook implementation artifacts, when possible to illustrate how our theory was applied in a real-life setting. We will then discuss the implementation in detail. Among other theoretical instruments, we will employ the pattern matching technique of data analysis proposed by Yin [157]. Following this technique we will observe how case study companies are using our handbook, identifying specific patterns, which we will then compare with the best practice patterns we propose in our theory. As a result we will study and report on the deviations between these patterns, discussing such deviations.

Finally, we will present our analysis on the predefined quality criteria for our theory evaluation. We will discuss these evaluation criteria for the implemented and studied sections of the handbook (representing parts of the proposed theory) and specific best practices in each of these sections. We will first report our analysis for the case studies individually, then discussing them side by side, presenting the common findings and differences across the cases. We will also discuss the effects and the shortcoming of using our theory, aiming at a critical theory evaluation as a result.

F

Handbook Implementation Artifacts

Appendix F presents some exemplary artifacts created and used by the employees at theory evaluation case studies at Companies A, B, and C. These artifacts were created during the guided implementation of the corporate open source governance handbook (a representation of the proposed theory). We did not influence the handbook implementation, but rather observed the implementation process (including the creation and use of such artifacts) with a goal of evaluating of our theory through multiple-case case study fol-

lowing the case study protocol presented in Appendix E.

F.1 Case Study A – Overview of FLOSS Governance Processes

As observed in our theory evaluation in Case Study A, this artifact was developed at Company A following our theory's best practices and their interconnections. They used our open source governance handbook (a practical representation of our theory) as the basis for developing this process overview. The artifact was developed and first used at Division A.1 of Company A with the goal of visually illustrating and communicating all the processes of corporate open source governance with the stakeholder employees. The employee (a developer from the R&D department at Division A.1) tasked with reading the full handbook and suggesting potential ways of implementing the proposed best practices and processes at Division A.1 at first, and then at the whole Company A created this artifact going beyond the original content of the handbook, expanding the suggested processes by adding company-specific IDs, color coding, and links between the best practices.

In this section of Appendix F we presented the corporate FLOSS governance processes on all the topics of the handbook. Figure F.1 covered the company processes for getting started with open source governance. Figure F.2 covered the company processes for general open source governance. Figure F.3 covered the company processes for inbound open source governance, focused on component approval. Figure F.4 covered the company processes for inbound open source governance, focused on component reuse. Figure F.5 covered the company processes for supply chain management related open source governance. Figure F.6 covered the company processes for outbound open source governance.

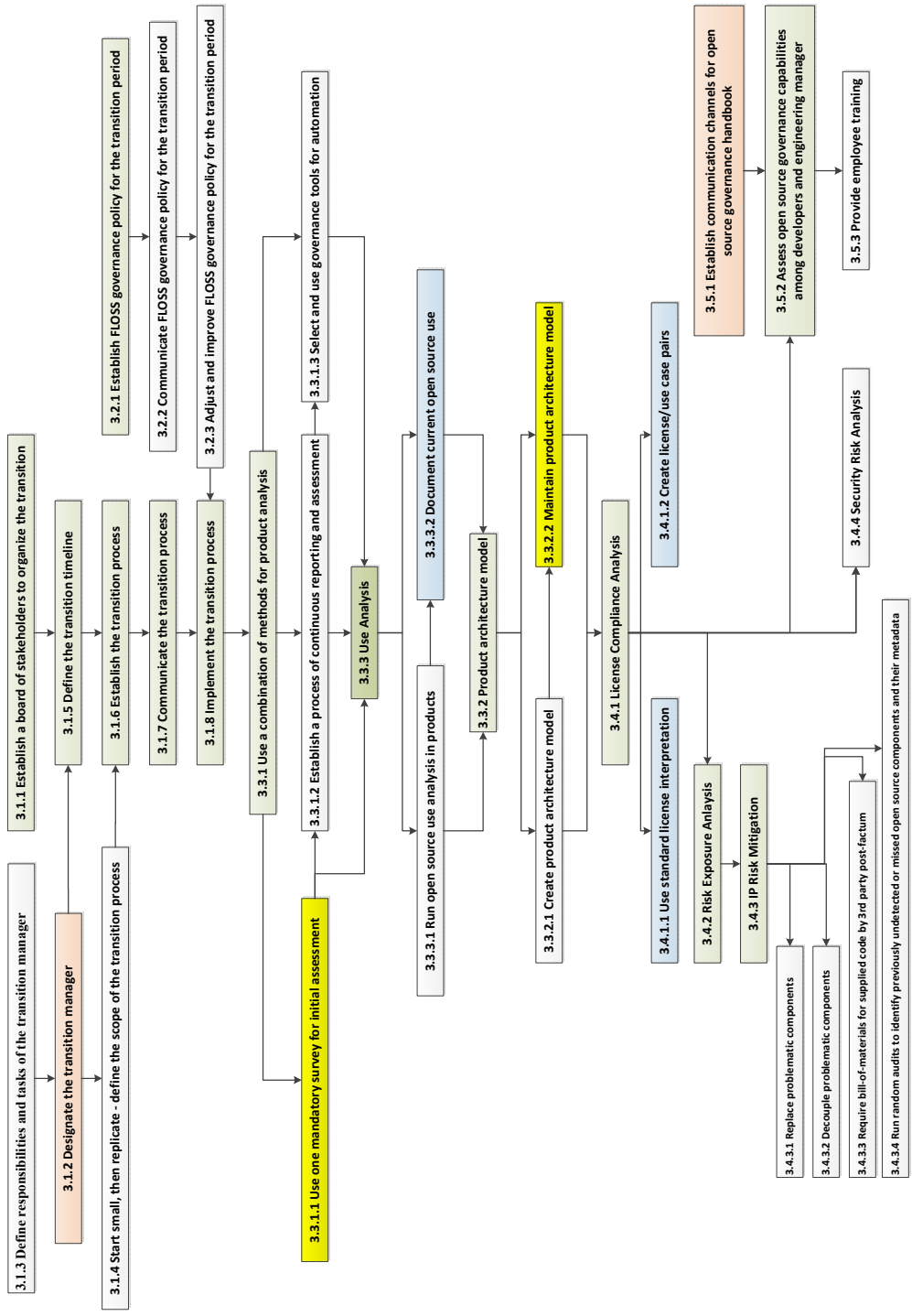


Figure F.1: Handbook Implementation Artifact at Case Study A – Overview of FLOSS Governance Processes on Getting Started

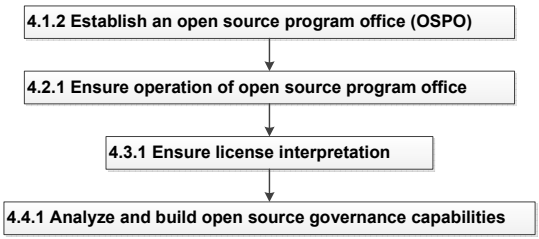


Figure F.2: Handbook Implementation Artifact at Case Study A – Overview of FLOSS Governance Processes on General Governance

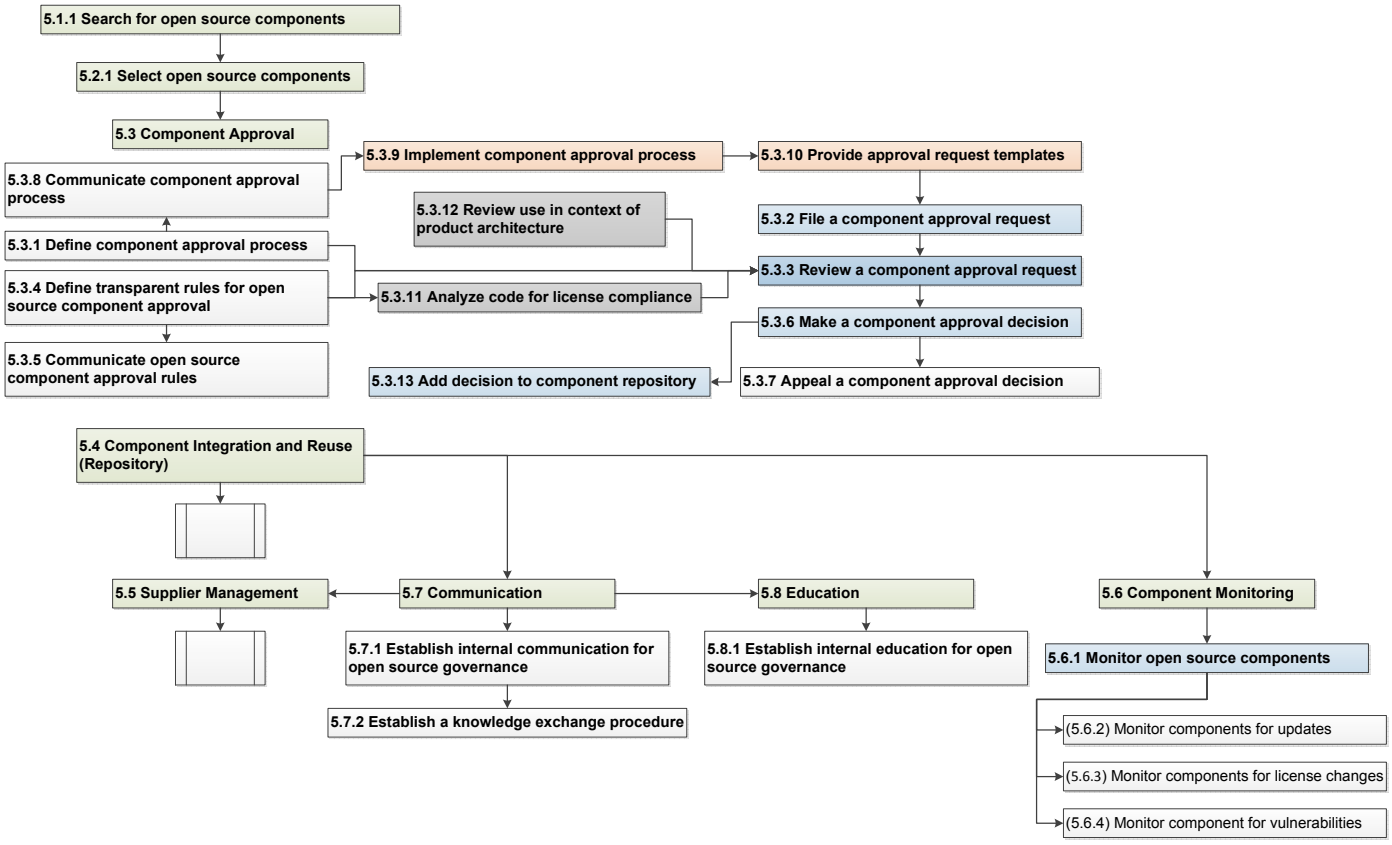


Figure F.3: Handbook Implementation Artifact at Case Study A – Overview of FLOSS Governance Processes on Inbound Governance, focused on Component Approval

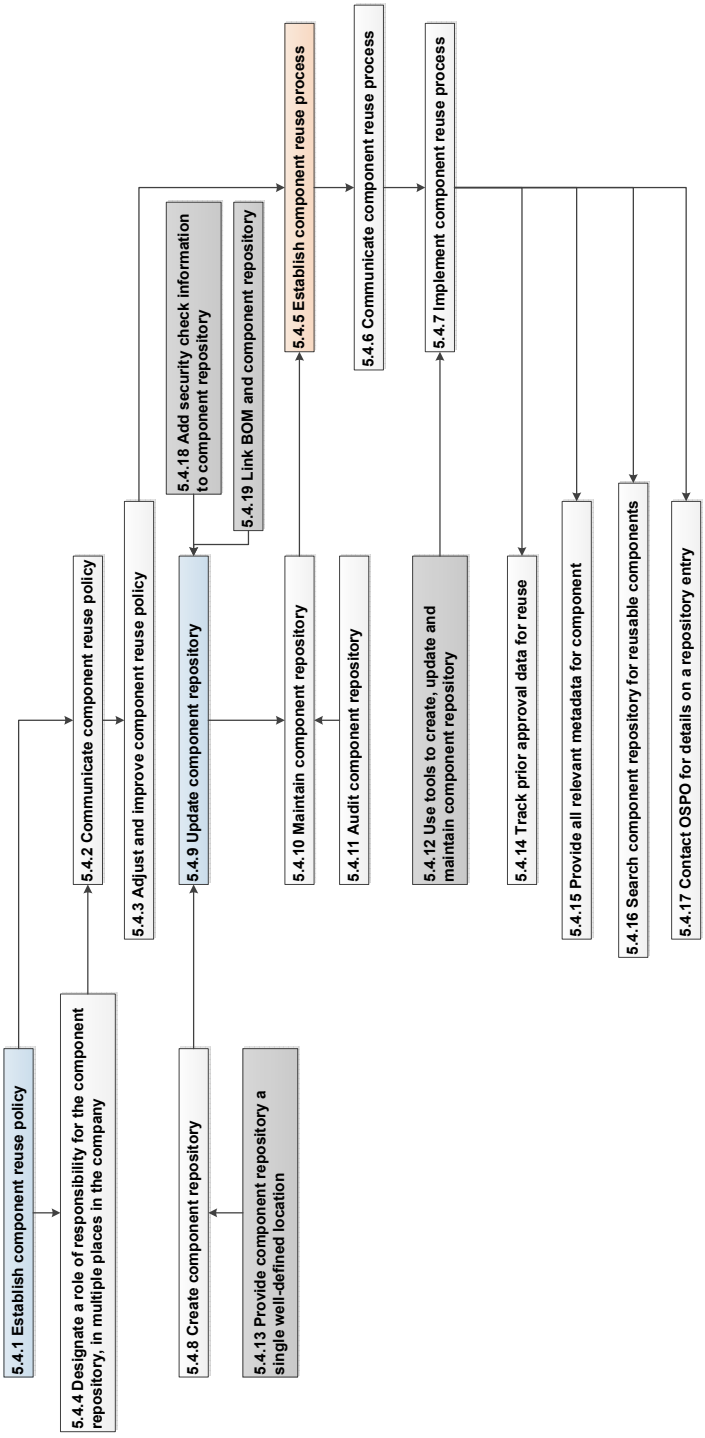


Figure F.4: Handbook Implementation Artifact at Case Study A – Overview of FLOSS Governance Processes on Inbound Governance, focused on Component Reuse

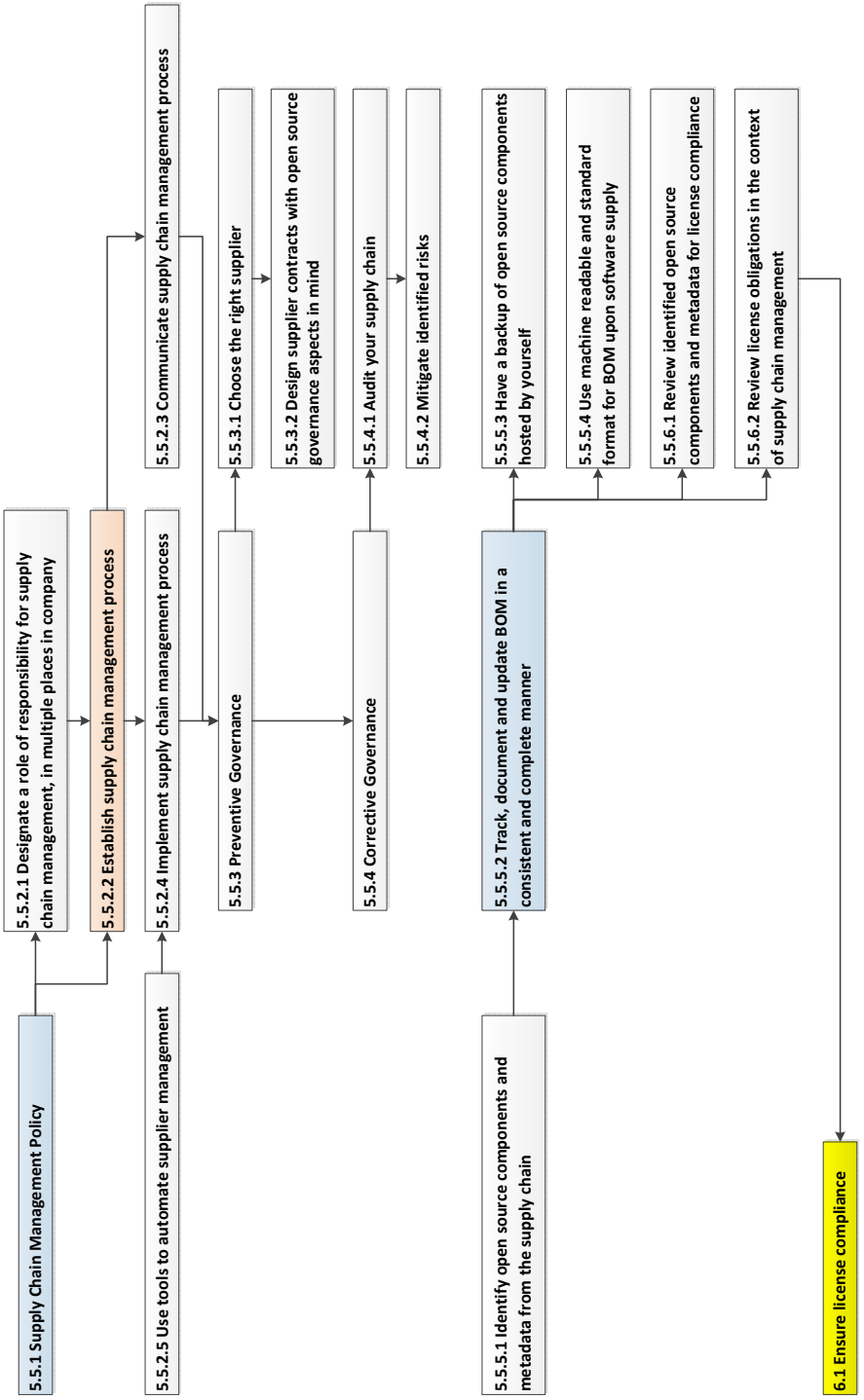


Figure F.5: Handbook Implementation Artifact at Case Study A – Overview of FLOSS Governance Processes on Supply Chain Management

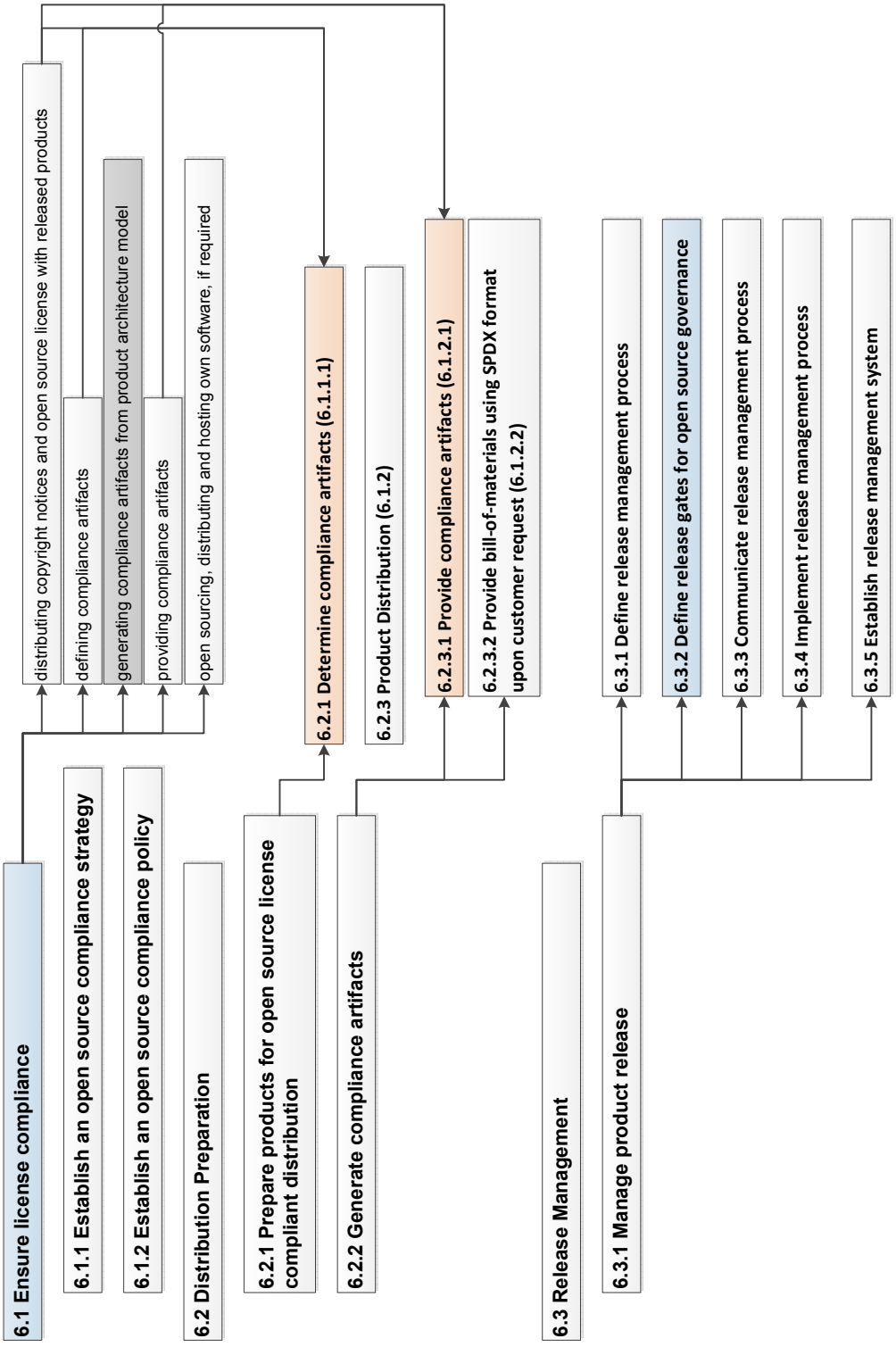


Figure F.6: Handbook Implementation Artifact at Case Study A – Overview of FLOSS Governance Processes on Outbound Governance

F.2 Case Study A – FOSSology Report Excerpt from Division A.1

As observed in our theory evaluation in Case Study A, this artifact was created at Company A following our theory's best practices for getting started A.3.9 (Run open source use analysis in products) and A.3.4 (Select and use governance tools for automation). The artifact was created after the first experimental run of open source use analysis in one product of Division A.1 of Company A, using the open source compliance tool FOSSology.

FOSSology

OSS Component Clearing Report [Excerpt] at Division A.1, Company A		
Clearing Information	**Department**	FOSSology Generation
	Prepared by	Employee X (employee_x)
	Reviewed by (opt.)	NA
	Report release date	2018/12/18
Component Information	**Community**	NA
	Component	NA
	Version	NA
	Component hash (SHA-1)	D2D346D1B90E4E1E0F7DB22CD92B6649DA7EF1C2
	Release date	NA
	Main license(s)	Main License(s) Not selected.
	Other license(s)	License(s) Not Identified.
	Fossology Upload/Package Link	http://nas02fra:8081/repo/?mod=showjobs&upload=5
	SW360 Portal Link	NA
	Result of License Scan	0BSD, AFL-2.0, AFL-2.1, [redacted], AMD, ATT, Apache, Apache-1.0, Apache-2.0, Artistic-1.0, Artistic-1.0-Perl, Artistic-2.0, Autoconf-exception, BSD, BSD-2-Clause, BSD-2-Clause-FreeBSD, BSD-2-Clause-NetBSD, BSD-3-Clause, BSD-4-Clause, BSD-4-Clause-UC, BSD-possibility, BSD-style, BSL-1.0, Bison-exception, Bison-exception-2.2, CC-BY-NC-SA-3.0, CC-BY-ND-2.0, CC-BY-SA, CC-BY-SA-3.0, CC0-1.0, CMU, CNRI-Python, ClArtistic, Cryptogams, DOC, Dual-license, FSF, FTL, Freeware, [redacted]

1. Assessment Summary

The following table only contains significant obligations, restrictions & risks for a quick overview – all obligations, restrictions & risks according to Section 3 must be considered.

General assessment	*NA*
Source / binary integration notes	☐ no critical files found, source code and binaries can be used as is ☐ critical files found, source code needs to be adapted and binaries possibly re-built
Dependency notes	☐ no dependencies found, neither in source code nor in binaries ☐ dependencies found in source code (see obligations) ☐ dependencies found in binaries (see obligations)
Export restrictions by copyright owner	☐ no export restrictions found ☐ export restrictions found (see obligations)
Restrictions for use (e.g. not for Nuclear Power) by copyright owner	☐ no restrictions for use found ☐ restrictions for use found (see obligations)
Additional notes	*NA*
General Risks (optional)	*NA*

2. Required license compliance tasks

2.1. Common obligations, restrictions and risks:

There is a list of common rules which was defined to simplify the To-Dos for development and distribution. The following list contains rules for development, and distribution which must always be followed!

2.1.1	**Documentation of license conditions and copyright notices in product documentation (License Notice File / README_OSS) is provided by this component clearing report:**
2.1.2	**Additional Common Obligations: Need to be ensured by the distributing party:**
2.1.3	**Obligations and risk assessment regarding distribution**

2.2. Additional obligations, restrictions & risks beyond common rules

This chapter contains all obligations in addition to "common obligations, restrictions and risks" (common rules) of included OSS licenses (need to get added manually during component clearing process).

Obligation	License	License section reference and short Description

3. Acknowledgements

(Reference to the license, Text of acknowledgements, File path)

4. Export Restrictions

The content of this paragraph is not the result of the evaluation of the export control experts (the ECCN). It contains information found by the scanner which shall be taken in consideration by the export control experts during the evaluation process. If the scanner identifies an ECCN it will be listed here. (NOTE: The ECCN is seen as an attribute of the component release and thus it shall be present in the component catalogue.

(Statements, Comments, File path)

5. Notes

Only such source code of this component may be used-

- which has been checked by and obtained via the Clearing Center or
- which has been submitted to Clearing Support to be checked

Other source code or binaries from the Internet **must not be** used.

The following chapters are generated by the source code scanner.

5.1. Notes on individual files

(License name, Comment Entered, File path)

FOSSology

6. Results of License Scan

(Scanner count, Concluded license count, License name)

99	0	0BSD
1	0	AFL-2.0
23	0	AFL-2.1
1	0	[redacted]
8	0	[redacted]
1	0	AMD
1	0	ATT
2	0	Apache
35	0	Apache-1.0
25	0	Apache-2.0
1031	0	Artistic-1.0
1	0	Artistic-1.0-Perl
2	0	Artistic-2.0
68	0	Autoconf-exception
404	0	BSD
66	0	BSD-2-Clause
12	0	BSD-2-Clause-FreeBSD
1	0	BSD-2-Clause-NetBSD
299	0	BSD-3-Clause
57	0	BSD-style
641	0	Dual-license
270	0	FSF
737	0	[redacted]
1690	0	MIT
[Excerpt; 4 pages removed]		

FOSSology

7. Main Licenses

(License name, License text, File path)

8. Other OSS Licenses (red) - specific obligations

(License name, License text, File path)

9. Other OSS Licenses (yellow) - additional obligations to common rules (e.g. copyleft)

(License name, License text, File path)

10. Other OSS Licenses (white) - only common rules

(License name, License text, File path)

11. Overview of All Licenses with or without Obligations

(License ShortName, Obligation)

12. Copyrights

(Statements, Comments, File path)

(C) COPYRIGHT 2000, Eric Busboom <eric@softwarestudio.org> This program is free software; you can redistribute it and/or modify it under the terms of either: The LGPL as published by the Free Software Foundation,	src/kernel.bz2/kernel/apalis-tk1/usr/include/libical/icalss.h

version 2.1, available at: http://www.gnu.org/licenses/lgpl-2.1.html Or: The Mozilla Public License Version 1.0. You may obtain a copy of the License at http://www.mozilla.org/MPL/ The Original Code is eric. The Initial Developer of the Original Code is Eric Busboom	
(C) COPYRIGHT 2000, Eric Busboom <eric@softwarestudio.org>	src/kernel.bz2/kernel/apalis-tk1/usr/include/libical/icalcalendar.h src/kernel.bz2/kernel/apalis-tk1/usr/include/libical/icalfileset.h src/kernel.bz2/kernel/apalis-tk1/usr/include/libical/icalmessage.h src/kernel.bz2/kernel/apalis-tk1/usr/include/libical/icalset.h
(C) 2010-2011 Red Hat, Inc.	src/kernel.bz2/kernel/apalis-tk1/usr/include/linux/netfilter/xt_AUDIT.h
(C) 2010 Red Hat Inc Author: Michael S. Tsirkin <mst@redhat.com>	src/kernel.bz2/kernel/apalis-tk1/usr/include/linux/netfilter/xt_CHECKSUM.h
(C) 2010 Nokia Corporation	src/kernel.bz2/kernel/apalis-tk1/usr/include/gstreamer-0.10/gst/pbutils/encoding-target.h src/kernel.bz2/kernel/apalis-tk1/usr/include/gstreamer-1.0/gst/pbutils/encoding-target.h
(C) 2010 Andras Timar <timar@fsf.hu>	src/kernel.bz2/kernel/apalis-tk1/usr/share/X11/xkb/symbols/hu
(C) 2009-2010 Nokia Corporation	src/kernel.bz2/kernel/apalis-tk1/usr/include/gstreamer-0.10/gst/pbutils/encoding-profile.h src/kernel.bz2/kernel/apalis-tk1/usr/include/gstreamer-1.0/gst/pbutils/encoding-profile.h
(C) 2009 Luotao Fu <l.fu@pengutronix.de>	src/kernel.bz2/kernel/apalis-tk1/usr/include/libsocketcan.h
(C) 2008 Wim Taymans <wim.taymans@gmail.com>	src/kernel.bz2/kernel/apalis-tk1/usr/include/gstreamer-1.0/gst/codecparsers/gsth264parser.h
(C) 2008 Hans de Goede <hdegoede@redhat.com>	src/kernel.bz2/kernel/apalis-tk1/usr/include/libv4l1.h src/kernel.bz2/kernel/apalis-tk1/usr/include/libv4l2.h src/kernel.bz2/kernel/apalis-tk1/usr/include/libv4lconvert.h
(C) 2005 Michal Benes <michal.benes@itonis.tv>	src/kernel.bz2/kernel/apalis-tk1/usr/include/gstreamer-1.0/gst/codecparsers/gsth264parser.h
(C) 2005 Jelmer Vernooij <jelmer@samba.org>	src/kernel.bz2/kernel/apalis-tk1/usr/share/perl5/Parse/Pidl/Samba4/COM/Header.pm
(C) 2005 Harald Welte <laforge@netfilter.org>	src/kernel.bz2/kernel/apalis-tk1/usr/include/linux/netfilter/xt_NFQUEUE.h
(C) 2004-2005, 2008 Jelmer Vernooij <jelmer@samba.org>	src/kernel.bz2/kernel/apalis-tk1/usr/share/perl5/Parse/Pidl/ODL.pm
(C) 2003 Andreas Gruenbacher <agruen@suse.de>	src/kernel.bz2/kernel/apalis-tk1/usr/include/linux/nfsacl.h
(C) 2003	src/kernel.bz2/kernel/apalis-tk1/usr/share/X11/xkb/symbols/ro
[redacted]	[redacted]
(C) 2002-2004 Soós Péter <sp@osb.hu>	src/kernel.bz2/kernel/apalis-tk1/usr/share/X11/xkb/symbols/hu
(C) 2002 by Harald Welte <laforge@gnumonks.org>	src/kernel.bz2/kernel/apalis-tk1/usr/include/linux/netfilter/xt_CHECKSUM.h src/kernel.bz2/kernel/apalis-tk1/usr/include/linux/netfilter_ipv4/ipt_ECN.h
[redacted]	[redacted]
(C) 2002 Harald Welte <laforge@gnumonks.org>	src/kernel.bz2/kernel/apalis-tk1/usr/include/linux/netfilter/xt_DSCP.h src/kernel.bz2/kernel/apalis-tk1/usr/include/linux/netfilter/xt_ecn.h
(C) 2001, 2004-2008.	src/kernel.bz2/kernel/apalis-tk1/usr/share/X11/xkb/symbols/ro

(C) 2001 Marc Boucher (marc@mbsi.ca).	src/kernel.bz2/kernel/apalis-tk1/usr/include/linux/netfilter/xt_conntrack.h
[redacted]	[redacted]
(C) 2000 by Harald Welte <laforge@netfilter.org> */	src/kernel.bz2/kernel/apalis-tk1/usr/include/linux/netfilter_ipv4/ipt_TTL.h
[excerpt; 1200+ pages removed]	

13. Bulk Findings

(License name, License text, File path)

14. Non Functional Licenses

In this section the files and their licenses can be listed which do not "go" into the delivered "binary", e.g. /test or /example.

15. Irrelevant Files

(Path, Files, Licenses)

15.1. Comment for Irrelevant files

(License name, Comment Entered, File path)

16. Clearing Protocol Change Log

Last Update	Responsible	Comments

F.3 Case Study B – Supplier Questionnaire on FLOSS Governance Maturity

As observed in our theory evaluation in Case Study B, this artifact was developed at Company B following our theory's best practices B.3.2 (Assess open source governance and compliance awareness and maturity) and B.3.3 (Request supplier certification or self-certification).

FOSS Compliance for Licensors

COMPANY is certified according to standard ISO norms. Therefore it requires its licensors to be compliant with the same norms or has to conduct regular supplier audits with its licensors so check whether the minimum standard processes are in place to ensure standard compliance, quality and security of the products.

1. Licensor Information

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

2. Quality Management

Use of QMS Practices	
Agile	[_] yes
SixSigma	[_] yes
TQM	[_] yes
other	specify:

QMS Certification Norms	Certification valid until:
ISO 9001	
AS 9101 or EN 9101	
ISO 13485	
ISO 16949	
others:	

3. Level of Open Source Awareness

How do you rank your Organization's awareness of Open Source by	good	basic	none
Management	[_]	[_]	[_]
Development Line Management	[_]	[_]	[_]
Developers	[_]	[_]	[_]
Sales	[_]	[_]	[_]
Legal Advisers	[_]	[_]	[_]
Do you follow standards like clearlydefined.io and OpenChain?	[_] yes		

4. Technologies used in the Development Process

Category	Description
Programming Languages	
Integrated Development Environment	
Source Code Versioning	
Build Tools	
Package/Dependency Managers	
Test Automation Tools	

5. Integration of Third Party Software into Product

Third Party Software comprises

- Open Source Software
- Free Software
- Shareware
- Commercially Licensed Software

Do you classify TP Components	**[_] yes**
using black/white List of Licenses	[_] yes
using black/white List for TP Components	[_] yes
using Lists of Vulnerabilities	[_] yes
Do you register Third Party Components	**[_] yes**
Manual Process	[_] yes
Automated Process	[_] yes
Application Support	[_] yes
Do you maintain a Bill of Material of Third Party Components used in your Product	[_] yes
Do you maintain Dependencies in your Bill of Material	[_] yes
Do you audit the use of TP Software	**[_] yes**
Source Code Audits	[_] yes
Binary Audits	[_] yes
Vulnerability Scans	[_] yes

6. Support of Sales Process

Are Customers informed about use of Third Party Software	[_] yes
Does License Agreement reflect Open Source Conditions	[_] yes
Is a Bill of Material provided to Customers	[_] yes
If yes: how and where:	
Is Source Code of Open Source provided to customers where required by Open Source License	[_] yes
If yes: how and where:	

7. Requirements to use Licensor's Products using Free and Open Source Software

a) Licensor shall provide Bill of Material in one of these suitable formats:

	Format	Specification
Minimum Requirement	PDF document	specs to be requested by COMPANY
Standard Requirement	Machine-readable `debian/copyright` file	https://www.debian.org/doc/packaging-manuals/copyright-format/1.0/
Best/ recommended Requirement	COMPANY specific SPDX Format	

b) Licensor shall provide Source Code of Open Source Components:

Minimum Requirement	where this is required by Open Source License (e.g. GPL family of licenses)
Standard Requirement	all components where Source Code is available
Descriptions of the Location: (FTP, GIT, ...)	

F.4 Case Study B – SPDX Requirements for Suppliers

As observed in our theory evaluation in Case Study B, this artifact was developed at Company B following our theory's best practices B.4.7 (Identify open source components and metadata from the supply chain) and B.4.10 (Use machine-readable and standard format for BOM upon software supply).

SPDX Requirements for Suppliers

Table of Contents

Introduction

Software Package Data Exchange (SPDX) is an open standard for communicating software bill of material (BOM) information including components, licenses, copyrights and security references.

Format

The SPDX document should be in the tag:value format which uses typically the suffix *.spdx. The document must be compliant with the SPDX Specification version 2.1.

Definitions

Term	Description
Supplier Package	Is the supplier's product e.g. Product XY 3.1
Third-party Package	Is the third-party product e.g. apache-ant 1.9.4

Mapping

Legend of cardinality:

Cardinality	Description
M	Mandatory
O	Optional
1	Exactly once
n	Zero or multiple times
1n	At least once

SPDX Tag	Cardinality Requirements	Comment
Document	M1	
SPDXVersion	M1	
DataLicense	M1	
SPDXID	M1	
DocumentName	M1	
DocumentNamespace	M1	
Creator	M1n	
Created	M1	

Supplier Package	M1n	
PackageName	M1	
PackageVersion	M1	
SPDXID	M1	
PackageCopyrightText	M1	
PackageDownloadLocation	M1	
PackageLicenseDeclared	M1	
PackageLicenseConcluded	M1	
FilesAnalyzed	O1	
Third-party Package	On	This section is required if third-party software is contained in the product specified in the Supplier Package
PackageName	M1	
PackageVersion	M1	
SPDXID	M1	
PackageCopyrightText	M1	
PackageDownloadLocation	M1	
PackageLicenseDeclared	M1	
PackageLicenseConcluded	M1	
FilesAnalyzed	O1	
License	On	This section is required if third-party software is licensed with a license that is not available on the SPDX license list at https://spdx.org/licenses/
LicenseID	M1	
LicenseName	M1	
ExtractedText	M1	
Relationship	M1n	
DESCRIBES	M1n	
PACKAGE_OF	On	

Example

F.5 Case Study B – Continuous Compliance Process

As observed in our theory evaluation in Case Study B, this artifact was developed at Company B following our theory's best practices B.4.12 (Review identified open source components and metadata for license compliance), B.4.13 (Review license obligations in the context of supply chain management), and other practices including OSGOV-INBGOV-COMAPP-1. Define the component approval process, presented in Table 3.9, and OSGOV-OUTGOV-LICCOM-1. Ensure license compliance, presented in Table 3.18.

This artifact illustrated a proposed compliance process, developed by a compliance officer at Company B following the above-mentioned best practices from our theory, and extending them aspiring to achieve continuous compliance. Our theory did not find this to be an industry best practice, as many of the interviewed experts during theory building deemed it to be unrealistic given the currently available compliance tools. Though we did find that companies want to have tools to meet a requirement of continuous compliance, as presented in our paper on industry requirements for FLOSS governance tools [68].

Figure F.7 presented one option of the proposed continuous compliance process at Company B. Figure F.8 extended this process by adding the governance tools (both internally developed and third-party ones). Figure F.9 completed the process by adding proposed auto-approval rules for process automation. Figure F.10 built upon the process by linking Company B's third-party (including open source) software component repository (TP Vault) to the public Maven repository. Figure F.11 extended the process by adding a central database for the third-party software component metadata.

The key legend for the above-mentioned figures included:

- *iData* – a Master Data Management system, which was used to manage Company B's product catalog. The catalog contained technical dependencies between different

products and their third-party products (TPP – third-party components including open source software).

- *PCI Scanner*, which identified requested (known) TPPs and gave as output the scanning results to be used to manage the BOM stored in iData repository.
- *TP Vault* – a Repository that contained requested TPPs (sources and binaries).
- *TPP Fetcher* – an internally developed tool that collected TPP metadata (component names, versions, licenses, copyrights, etc.) from different sources within the built environment. Sources could be dependency managers/declarations or source code scans. It fetched TPP files (source code and binaries) that belonged to a TPP via package managers. It uploaded TPP metadata and TPP files to the TPP Interface.
- *TPP Interface* – an internally developed tool that took TPP metadata to create requests and uploaded TPP files to TP Vault, which triggered the TPP review process.

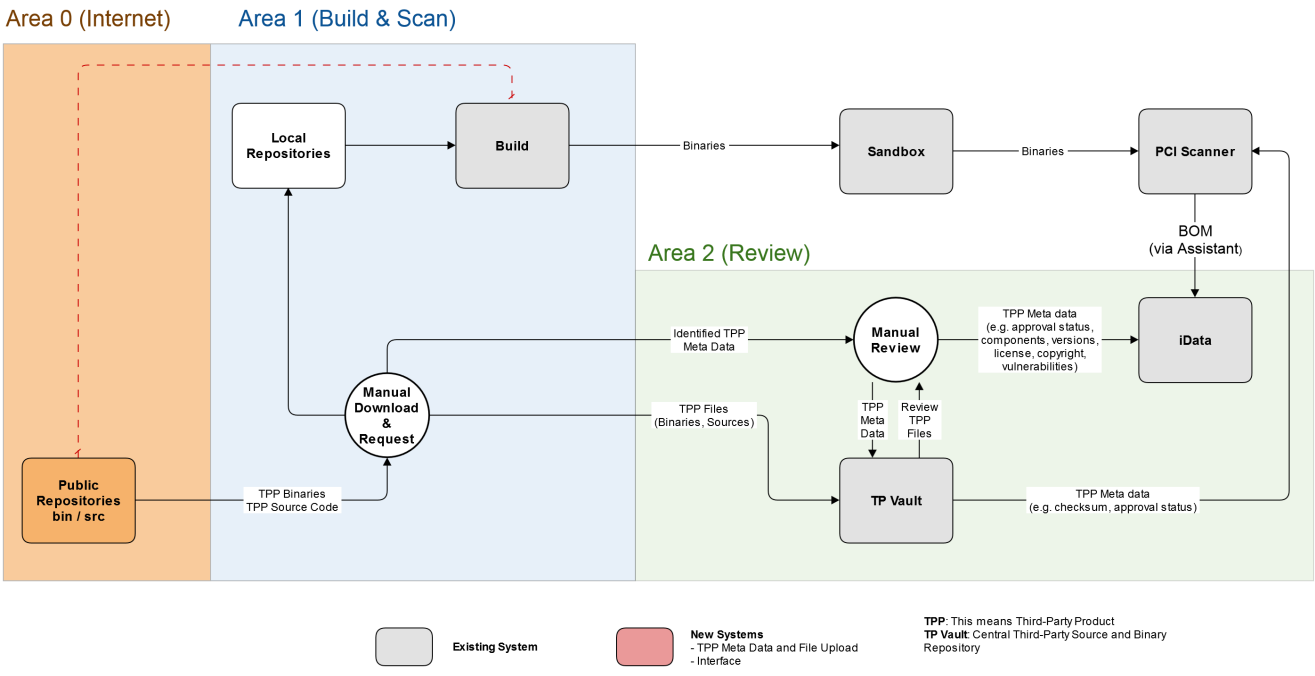


Figure F.7: Handbook Implementation Artifact at Case Study B – Continuous Compliance Process Version 1

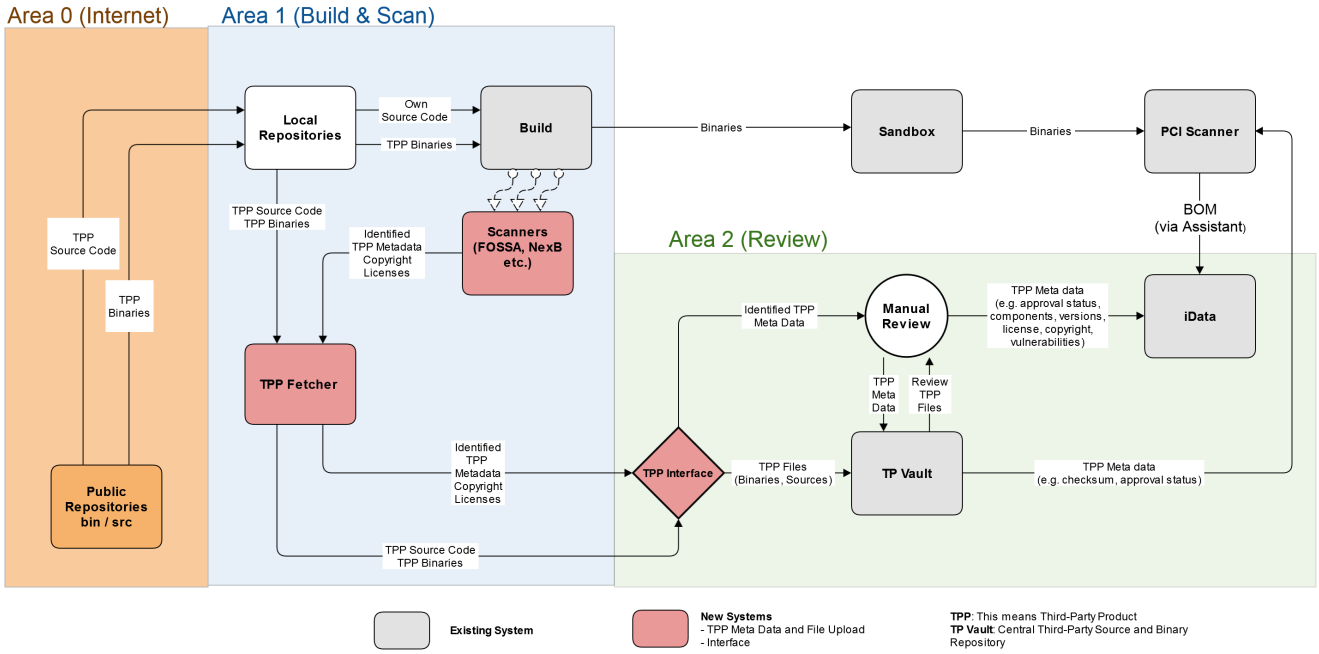


Figure F.8: Handbook Implementation Artifact at Case Study B – Continuous Compliance Process Version 2

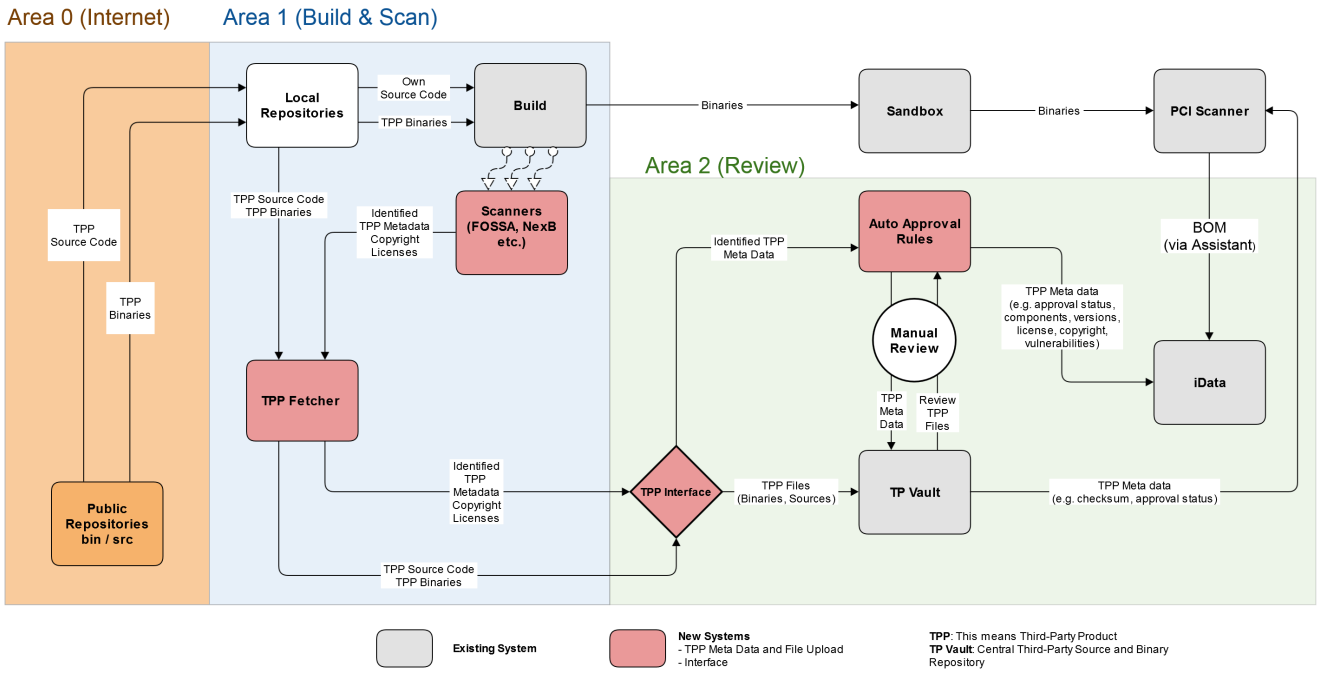


Figure F.9: Handbook Implementation Artifact at Case Study B – Continuous Compliance Process Version 3

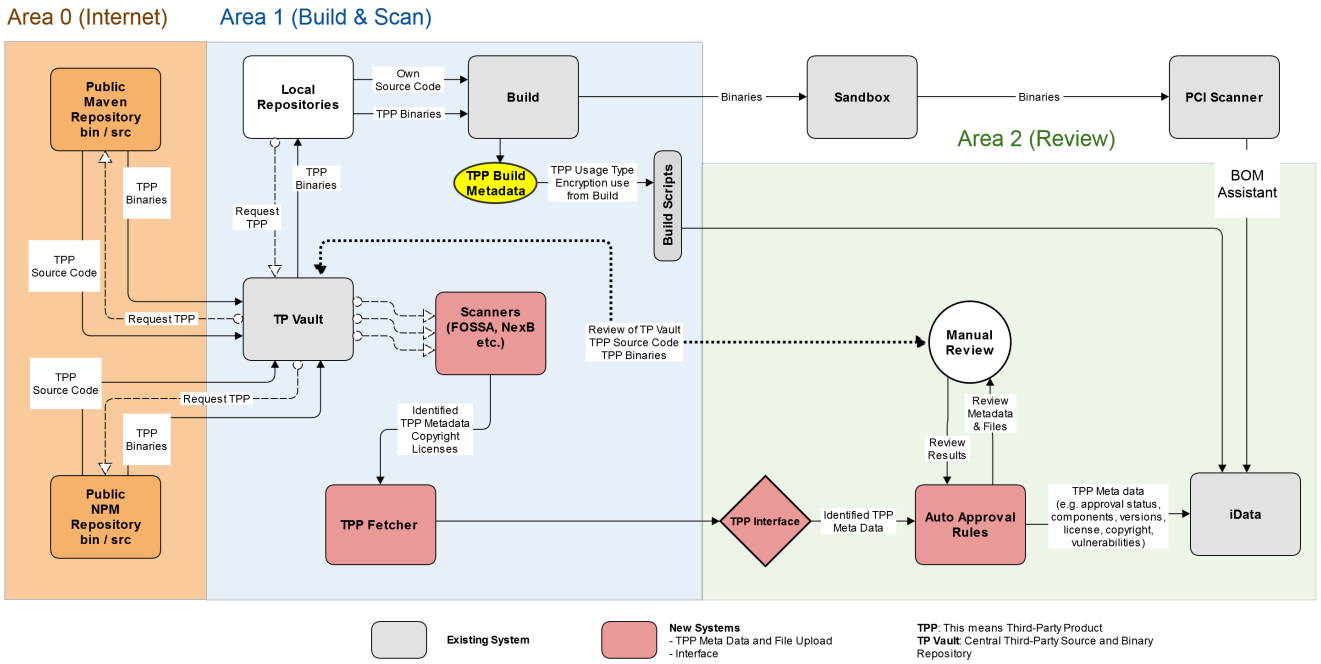


Figure F.10: Handbook Implementation Artifact at Case Study B – Continuous Compliance Process Version 4

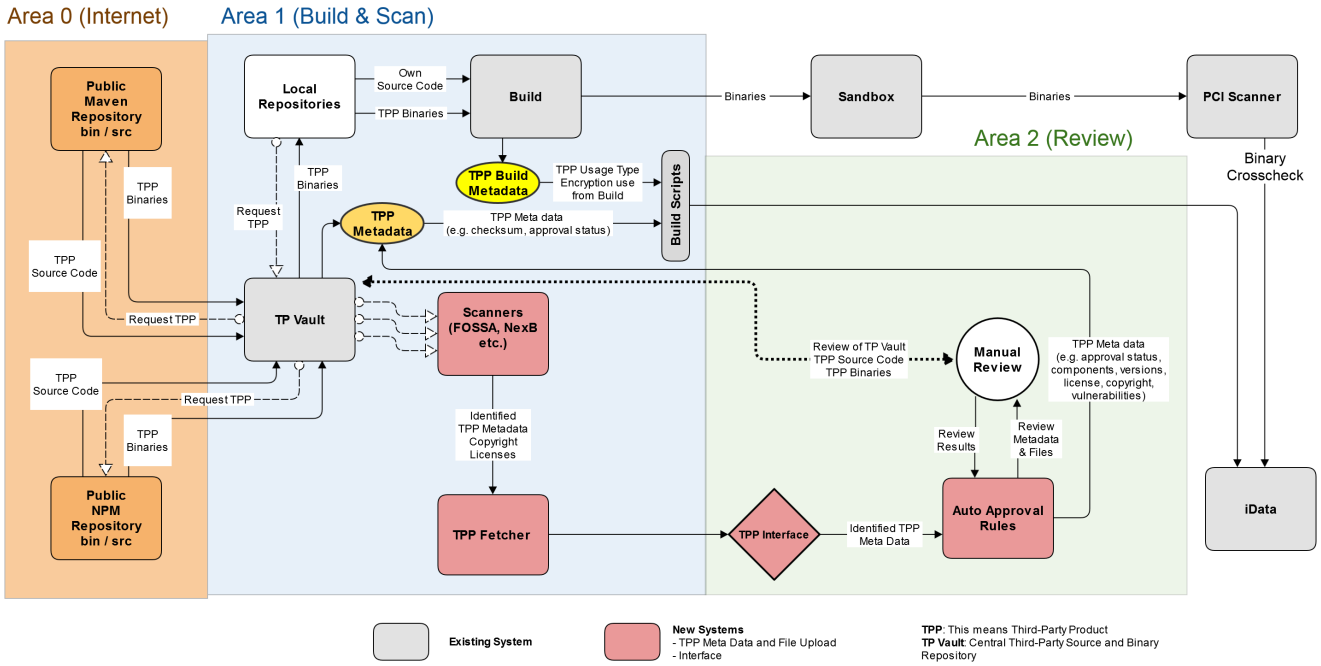


Figure F.11: Handbook Implementation Artifact at Case Study B – Continuous Compliance Process Version 5

F.6 Case Study C – Tooling for FLOSS Governance and Compliance

As observed in our limited theory evaluation in Case Study C, this artifact was used at Company C following our theory's best practices A.3.4 (Select and use governance tools for automation) and OSGOV-INBGOV-COMREU-12. Use tools to create, update and maintain component repository (for tools focused on component reuse).

Fossology

Fossology is one of the popular tools for OSS compliance and focuses on detection of licenses, copyrights, or export controls information. The basic workflow using Fossology is dividable in the following steps:

License scanning

To detect licenses, copyrights, or export controls Fossology using different pattern recognition methods.

License review/clearing

One of the key features of FOSSology is the user interface to review license findings in order to determine the exact licensing of a file. A review of the findings is necessary because the unequivocal detection of a license is not trivial. For instance, the license text can be modified or a complete unknown license is detected. The review/clearing results are stored in a database and can be reused for other scans.

Report generation

In the last step, a report with all detected components with their licenses, copyright, and export restriction information can be created. This list is called bill-of-materials (BOM). The usual representation of a BOM is in the form of the machine-readable SPDX standard.

SW360

SW360 on the other hand, helping users by establishing a central hub for software components in an organization. SW360 allows for

- tracking components used by a project/product,
- assessing security vulnerabilities,
- maintaining license obligations,
- enforcing policies, and
- generating legal documents.
- Integration with other tools and data sources (e.g. license scanner, static code analysis, build infrastructure, etc.)

SW360 doesn't provide necessary functionalities for license clearing by itself; instead, it can trigger a clearing process in FOSSology and import the resulting clearing reporting.

The is another example where different compliance tools can be used together, or depend on each other, e.g., OSS Review Toolkit (ORT) and ScanCode.

ScanCode from NexB

ScanCode is another popular exemplar of a license scanner like Fossology, with similar functionalities. No further explanation here because it's very similar to Fossology.

OSS Review Toolkit

The goal of the OSS Review Toolkit (ORT) is to verify Free and Open Source Software license compliance by checking project source code and dependencies. ORT analyzing the source code for dependencies, downloading the source code of the dependencies, scanning all source code for license information, and summarizing the results. It uses data from a projects build system (Maven, Gradle, etc.) to determen all components and it's dependencies and for the actual license check it triggers one of four supported license scanner. ScanCode is here the recommended option. ORT also using SPDX as a format to exchange results with other tools.

In parallel to using this tool comparison, Company C also used the list of the common industry requirements for open source governance and compliance tools resulting from our previous research [68]. Excerpts from this list of requirements are presented in Figure F.12 – tool requirements for tracking and reuse of FLOSS components, and in Figure F.13 – tool requirements for license compliance of FLOSS components.

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.

Figure F.12: Handbook Implementation Artifact at Case Study C – Tool Requirements for Tracking and Reuse of FLOSS Components

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.

Figure F.13: Handbook Implementation Artifact at Case Study C – Tool Requirements for License Compliance of FLOSS Components

References

[1] Agerfalk, P. J., Deverell, A., Fitzgerald, B., & Morgan, L. (2006). 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.

[2] Akkanen, J., Demeter, H., Eppel, T., Iványfi, Z., Nurminen, J. K., & Stenman, P. (2007). Reusing an open source application—practical experiences with a mobile crm pilot. In *IFIP International Conference on Open Source Systems* (pp. 217–222).: Springer.

[3] Aksulu, A. & Wade, M. (2010). A comprehensive review and synthesis of open source research. *Journal of the Association for Information Systems*, 11(11).

[4] Alspaugh, T. A., Asuncion, H. U., & Scacchi, W. (2009). 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.

[5] Andersen-Gott, M., Ghinea, G., & Bygstad, B. (2012). Why do commercial companies contribute to open source software? *International Journal of Information Management*, 32(2), 106–117.

[6] Ardagna, C. A., Banzi, M., Damiani, E., & Frati, F. (2010). Implementing open source software governance in real software assurance processes. In *International Conference of Software Business* (pp. 103–114).: Springer.

[7] Ayala, C., Hauge, Ø., Conradi, R., Franch, X., Li, J., & Velle, K. S. (2009). Challenges of the open source component marketplace in the industry. In *IFIP International Conference on Open Source Systems* (pp. 213–224).: Springer.

[8] Ayala, C. P., Cruzes, D., Hauge, Ø., & Conradi, R. (2011). Five facts on the adoption of open source software. *IEEE Software*, 28(2), 95–99.

[9] Baskerville, R. L. & Wood-Harper, A. T. (1996). A critical perspective on action research as a method for information systems research. *Journal of Information Technology*, 11(3), 235–246.

[10] Beamon, B. M. (2008). Sustainability and the future of supply chain management. *Operations and Supply Chain Management*, 1(1), 4–18.

[11] Beck, C. T. (1993). Qualitative research: The evaluation of its credibility, fittingness, and auditability. *Western Journal of Nursing Research*, 15(2), 263–266.

[12] Berglund, E. & Priestley, M. (2001). Open-source documentation: in search of user-driven, just-in-time writing. In *Proceedings of the 19th Annual International Conference on Computer Documentation* (pp. 132–141).: ACM.

[13] Bitsch, V. (2005). Qualitative research: A grounded theory example and evaluation criteria. *Journal of Agribusiness*, 23(345-2016-15096).

[14] Bitzer, J., Schrettl, W., & Schröder, P. J. (2007). Intrinsic motivation in open source software development. *Journal of Comparative Economics*, 35(1), 160–169.

[15] Blecken, A. & Hellingrath, B. (2008). Supply chain management software for humanitarian operations: review and assessment of current tools. *Proceedings of the 5th ISCRAM*, (pp. 342–351).

[16] Boldyreff, C., Nutter, D., Rank, S., et al. (2002). Architectural requirements for an open source component and artefact repository system within genesis. In *Open Source Software Development Workshop*.

[17] Bonaccorsi, A., Giannangeli, S., & Rossi, C. (2006). Entry strategies under competing standards: Hybrid business models in the open source software industry. *Management Science*, 52(7), 1085–1098.

[18] Bonaccorsi, A. & Rossi, C. (2003). Why open source software can succeed. *Research Policy*, 32(7), 1243–1258.

[19] Brown, A. W. & Booch, G. (2002). Reusing open source software and practices: The impact of open-source on commercial vendors. In *International Conference on Software Reuse* (pp. 123–136).: Springer.

[20] Calder, B. J., Phillips, L. W., & Tybout, A. M. (1982). The concept of external validity. *Journal of Consumer Research*, 9(3), 240–244.

[21] Capek, P. G., Frank, S. P., Gerdt, S., & Shields, D. (2005). A history of ibm's open-source involvement and strategy. *IBM Systems Journal*, 44(2), 249–257.

[22] Capra, E., Francalanci, C., & Merlo, F. (2008). An empirical study on the relationship between software design quality, development effort and governance in open source projects. *IEEE Transactions on Software Engineering*, 34(6), 765–782.

[23] Capraro, M. & Riehle, D. (2017). Inner source definition, benefits, and challenges. *ACM Computing Surveys (CSUR)*, 49(4), 67.

[24] Cavaye, A. L. (1996). Case study research: a multi-faceted research approach for is. *Information Systems Journal*, 6(3), 227–242.

[25] Chang, S., Lee, J., & Yi, W. (2010). A practical management framework for commercial software development with open sources. In *2010 IEEE 7th International Conference on E-Business Engineering* (pp. 164–171).: IEEE.

[26] Chau, P. Y. & Tam, K. Y. (1997). Factors affecting the adoption of open systems: an exploratory study. *MIS Quarterly*, (pp. 1–24).

[27] Chen, W., Li, J., Ma, J., Conradi, R., Ji, J., & Liu, C. (2008). An empirical study on software development with open source components in the chinese software industry. *Software Process: Improvement and Practice*, 13(1), 89–100.

[28] Clements, P. C., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Nord, R., & Stafford, J. (2004). A practical method for documenting software architectures.

[29] Conlon, P. & Carew, P. (2005). A risk driven framework for open source information systems development. In *1st International Conference on Open Source Systems* (pp. 200–203).

[30] Cook, T. D., Campbell, D. T., & Peracchio, L. (1990). *Quasi Experimentation*. Consulting Psychologists Press.

[31] Copenhaver, K. (2010). Open source policies and processes for inbound software. *International Free and Open Source Software Law Review*, 1(2), 143–154.

[32] Coughlan, S. (2017). The bid by openchain to transform the supply chain. *IFOSS L. Rev.*, 9, 45.

[33] Coughlan, S., Noda, T., & Tansho, T. (2013). A case study of the collaborative approaches to sustain open source business models. In *Proceedings of the 9th International Symposium on Open Collaboration*: ACM.

[34] Crowston, K., Wei, K., Howison, J., & Wiggins, A. (2012). Free/libre open-source software development: What we know and what we do not know. *ACM Computing Surveys (CSUR)*, 44(2).

[35] Dahlander, L. & Magnusson, M. G. (2005). Relationships between open source software companies and communities: Observations from nordic firms. *Research Policy*, 34(4), 481–493.

[36] Dedrick, J. & West, J. (2003). Why firms adopt open source platforms: a grounded theory of innovation and standards adoption. In *Proceedings of the Workshop on Standard Making: A Critical Research Frontier for Information Systems* (pp. 236–257).

[37] Dedrick, J. & West, J. (2004). An exploratory study into open source platform adoption. In *Proceedings of the 37th Annual Hawaii International Conference on System Sciences*: IEEE.

[38] Denzin, N. K. & Lincoln, Y. S. (2011). *The Sage Handbook of Qualitative Research*. Sage.

[39] Deodhar, S. J., Saxena, K., & Ruohonen, M. (2010). Firm-oriented success factors of an open source software (oss) product. In *Proceedings of the 3rd International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development* (pp. 1–4).: ACM.

[40] Deshpande, A. & Riehle, D. (2008). The total growth of open source. In *IFIP International Conference on Open Source Systems* (pp. 197–209).: Springer.

[41] Dohrn, H. & Riehle, D. (2011). Design and implementation of the sweble wikitext parser: unlocking the structured data of wikipedia. In *Proceedings of the 7th International Symposium on Wikis and Open Collaboration* (pp. 72–81).: ACM.

[42] Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of Management Review*, 14(4), 532–550.

[43] Enkel, E., Gassmann, O., & Chesbrough, H. (2009). Open r&d and open innovation: exploring the phenomenon. *R&d Management*, 39(4), 311–316.

[44] Fendt, O., Jaeger, M., & Serrano, R. J. (2016). Industrial experience with open source software process management. In *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, volume 2 (pp. 180–185).: IEEE.

[45] Fink, A. (2003). *The survey handbook*. Sage Publications.

[46] Fitzgerald, B. (2006). The transformation of open source software. *MIS Quarterly*, (pp. 587–598).

[47] Franch Gutiérrez, J., Susi, A., Annosi, M. C., Ayala Martínez, C. P., Glott, R., Gross, D., Kenett, R., Mancinelli, F., Ramsany, P., Thomas, C., et al. (2013). Managing risk in open source software adoption. In *Proceedings of the 8th International Joint Conference on Software Technologies (ICSOFT 2013)* (pp. 258–264).

[48] Francis, J. J., Johnston, M., Robertson, C., Glidewell, L., Entwistle, V., Eccles, M. P., & Grimshaw, J. M. (2010). What is an adequate sample size? operationalising data saturation for theory-based interview studies. *Psychology and Health*, 25(10), 1229–1245.

[49] Gamalielsson, J. & Lundell, B. (2016). On involvement in open standards: How do organisations contribute to w3c standards through editorship? In *IFIP International Conference on Open Source Systems* (pp. 57–70).: Springer.

[50] Gamalielsson, J. & Lundell, B. (2017). On licensing and other conditions for contributing to widely used open source projects: an exploratory analysis. In *Proceedings of the 13th International Symposium on Open Collaboration*: ACM.

[51] Gandhi, R., Germonprez, M., & Link, G. J. (2018). Open data standards for open source software risk management routines: An examination of spdx. In *Proceedings of the 2018 ACM Conference on Supporting Groupwork* (pp. 219–229).: ACM.

[52] Gangadharan, G., D'Andrea, V., De Paoli, S., & Weiss, M. (2012). Managing license compliance in free and open source software development. *Information Systems Frontiers*, 14(2), 143–154.

[53] German, D. & Di Penta, M. (2012). A method for open source license compliance of java applications. *IEEE Software*, 29(3), 58–63.

[54] German, D. M. & Hassan, A. E. (2009). 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.

[55] Germonprez, M., Young, B., Mathiassen, L., Kendall, J. E., Kendall, K. E., Warner, B., & Cao, L. (2012). Risk mitigation in corporate participation with open source communities: protection and compliance in an open source supply chain. *Risk*, 12, 15–2012.

[56] Gibbert, M., Ruigrok, W., & Wicki, B. (2008). What passes as a rigorous case study? *Strategic Management Journal*, 29(13), 1465–1474.

[57] Glynn, E., Fitzgerald, B., & Exton, C. (2005). Commercial adoption of open source software: an empirical study. In *2005 International Symposium on Empirical Software Engineering*: IEEE.

[58] Gobeille, R. (2008). The fossology project. In *Proceedings of the International Working Conference on Mining Software Repositories* (pp. 47–50).: ACM.

[59] Goertz, G. & Mahoney, J. (2012). *A tale of two cultures: Qualitative and quantitative research in the social sciences*. Princeton University Press.

[60] Gordon, T. F. (2011). Analyzing open source license compatibility issues with carneades. In *Proceedings of the 13th International Conference on Artificial Intelligence and Law* (pp. 51–55).: ACM.

[61] Guba, E. G. (1981). Criteria for assessing the trustworthiness of naturalistic inquiries. *Ectj*, 29(2).

[62] Guest, G., Bunce, A., & Johnson, L. (2006). How many interviews are enough? an experiment with data saturation and variability. *Field Methods*, 18(1), 59–82.

[63] Hammouda, I., Mikkonen, T., Oksanen, V., & Jaaksi, A. (2010). Open source legality patterns: architectural design decisions motivated by legal concerns. In *Proceedings of the 14th International Academic MindTrek Conference: Envisioning Future Media Environments* (pp. 207–214).: ACM.

[64] Hannebauer, C. & Gruhn, V. (2019). An open source pattern language. In *Transactions on Pattern Languages of Programming IV* (pp. 76–99). Springer.

[65] Hannebauer, C., Link, C., & Gruhn, V. (2014). Patterns for the distribution of power in floss projects. In *Proceedings of the 19th European Conference on Pattern Languages of Programs*: ACM.

[66] Hannebauer, C., Wolff-Marting, V., & Gruhn, V. (2010). Towards a pattern language for floss development. In *Proceedings of the 17th Conference on Pattern Languages of Programs*: ACM.

[67] Hannebauer, C., Wolff-Marting, V., & Gruhn, V. (2011). Contributor-interaction patterns in floss development. In *Proceedings of the 16th European Conference on Pattern Languages of Programs*: ACM.

[68] Harutyunyan, N., Bauer, A., & Riehle, D. (2018). Understanding industry requirements for floss governance tools. In I. Stamelos, J. M. Gonzalez-Barahoña, I. Varlamis, & D. Anagnostopoulos (Eds.), *IFIP International Conference on Open Source Systems* (pp. 151–167).: Springer.

[69] Harutyunyan, N., Bauer, A., & Riehle, D. (2019). Industry requirements for floss governance tools to facilitate the use of floss components in commercial products. *Journal of Systems and Software: Under Review*.

[70] Harutyunyan, N. & Riehle, D. (2019a). Getting started with floss governance and compliance: A theory of industry best practices. In *Proceedings of the 15th International Symposium on Open Collaboration*: Forthcoming.

[71] Harutyunyan, N. & Riehle, D. (2019b). Industry best practices for floss governance and component reuse. In *Proceedings of the 23rd European Conference on Pattern Languages of Programs*: Forthcoming.

[72] Hauge, Ø., Ayala, C., & Conradi, R. (2010). Adoption of open source software in software-intensive organizations–a systematic literature review. *Information and Software Technology*, 52(11), 1133–1154.

[73] Hauge, Ø., Sørensen, C.-F., & Conradi, R. (2008). Adoption of open source in the software industry. In *IFIP International Conference on Open Source Systems* (pp. 211–221).: Springer.

[74] Helmreich, M. (2011). *Best practices of adopting open source software in closed source software products*. PhD thesis, Diplomarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg.

[75] von Hippel, E. & von Krogh, G. (2003). Open source software and the "private-collective" innovation model: Issues for organization science. *Organization Science*, 14(2), 209–223.

[76] Jaaksi, A. (2007). Experiences on product development with open source software. In *IFIP International Conference on Open Source Systems* (pp. 85–96).: Springer.

[77] Jansen, H. (2010). 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).

[78] Jansen, S., Finkelstein, A., & Brinkkemper, S. (2009). A sense of community: A research agenda for software ecosystems. In *2009 31st International Conference on Software Engineering-Companion Volume* (pp. 187–190).: IEEE.

[79] Jiang, Q., Qin, J., & Kang, L. (2015). A literature review for open source software studies. In *International Conference on HCI in Business* (pp. 699–707).: Springer.

[80] Kapitsaki, G. M., Tselikas, N. D., & Foukarakis, I. E. (2015). An insight into license tools for open source software systems. *Journal of Systems and Software*, 102, 72–87.

[81] Kaufmann, A. & Riehle, D. (2017). The qdacity-re method for structural domain modeling using qualitative data analysis. *Requirements Engineering*, (pp. 1–18).

[82] Kemp, R. (2009). Towards free/libre open source software governance in the organization. *IFOSS L. Rev.*, 1.

[83] Kogut, B. & Metiu, A. (2001). Open-source software development and distributed innovation. *Oxford Review of Economic Policy*, 17(2), 248–264.

[84] Koltun, P. (2011). Free and open source software compliance: An operational perspective. *IFOSS L. Rev.*, 3.

[85] Krefting, L. (1991). Rigor in qualitative research: The assessment of trustworthiness. *The American Journal of Occupational Therapy*, 45(3), 214–222.

[86] Krivoruchko, J. (2007). The use of open source software in enterprise distributed computing environments. In *IFIP International Conference on Open Source Systems* (pp. 277–282).: Springer.

[87] von Krogh, G. & Spaeth, S. (2007). The open source software phenomenon: Characteristics that promote research. *The Journal of Strategic Information Systems*, 16(3), 236–253.

[88] von Krogh, G. & von Hippel, E. (2006). The promise of research on open source software. *Management Science*, 52(7), 975–983.

[89] Kuan, J. (2002). Open source software as lead user's make or buy decision: a study of open and closed source quality. *Stanford Institute for Economic Policy Research, Stanford University*.

[90] Lau, K.-K. & Wang, Z. (2007). Software component models. *IEEE Transactions on Software Engineering*, 33(10), 709–724.

[91] Lee, S.-Y. T., Kim, H.-W., & Gupta, S. (2009). Measuring open source software success. *Omega*, 37(2), 426–438.

[92] Leonard, L. (2010). Floss strategic thinking: a proposed framework to support strategic decision for commercial open source companies. In *4th FLOSS International Workshop on Free/Libre Open Source Software, Jena, Germany*.

[93] Lerner, J. & Tirole, J. (2005). The economics of technology sharing: Open source and beyond. *Journal of Economic Perspectives*, 19(2), 99–120.

[94] Li, J., Conradi, R., Slyngstad, O. P., Torchiano, M., Morisio, M., & Bunse, C. (2008). A state-of-the-practice survey of risk management in development with off-the-shelf software components. *IEEE Transactions on Software Engineering*, 34(2), 271–286.

[95] Li, Y., Tan, C.-H., & Teo, H.-H. (2012). Leadership characteristics and developers' motivation in open source software development. *Information & Management*, 49(5), 257–267.

[96] Lin, L. C.-H. & Shen, N. (2019). Copyleft referring to gpl-3.0 was cited as a defense method in chinese intellectual property court in beijing. *International Free and Open Source Software Law Review*, 10(1), 1–7.

[97] Lincoln, Y. S. & Guba, E. G. (1985). Establishing trustworthiness. *Naturalistic inquiry*, 289.

[98] Link, C. (2010). Patterns for the commercial use of open source: legal and licensing aspects. In *Proceedings of the 15th European Conference on Pattern Languages of Programs*: ACM.

[99] Lombard, M., Snyder-Duch, J., & Bracken, C. C. (2002). Content analysis in mass communication: Assessment and reporting of intercoder reliability. *Human Communication Research*, 28(4), 587–604.

[100] López, L., Costal, D., Ayala, C. P., Franch, X., Annosi, M. C., Glott, R., & Haaland, K. (2015). Adoption of oss components: a goal-oriented approach. *Data & Knowledge Engineering*, 99, 17–38.

[101] Lovejoy, J., Odence, P., & Lamons, S. (2013). Advancing the software package data exchange: An update on spdx. *International Free and Open Source Software Law Review*, 5(2), 145–152.

[102] Lundell, B. & Gamalielsson, J. (2013). Open standards and open source in swedish schools: On promotion of openness and transparency. In *IFIP International Conference on Open Source Systems* (pp. 207–221).: Springer.

[103] Lundell, B., Lings, B., & Lindqvist, E. (2006). Perceptions and uptake of open source in swedish organisations. In *IFIP International Conference on Open Source Systems* (pp. 155–163).: Springer.

[104] Lundell, B., Lings, B., & Lindqvist, E. (2010). Open source in swedish companies: where are we? *Information Systems Journal*, 20(6), 519–535.

[105] Madanmohan, T. et al. (2004). Notice of violation of ieee publication principles open source reuse in commercial firms. *IEEE Software*, 21(6), 62–69.

[106] Mancinelli, F., Boender, J., Di Cosmo, R., Vouillon, J., Durak, B., Leroy, X., & Treinen, R. (2006). Managing the complexity of large free and open source package-based software distributions. In *21st IEEE/ACM International Conference on Automated Software Engineering (ASE'06)* (pp. 199–208).: IEEE.

[107] Markus, M. L. (2007). The governance of free/open source software projects: monolithic, multidimensional, or configurational? *Journal of Management & Governance*, 11(2), 151–163.

[108] Mateos-Garcia, J. & Steinmueller, W. E. (2008). The institutions of open source software: Examining the debian community. *Information Economics and Policy*, 20(4), 333–344.

[109] Mays, N. & Pope, C. (1995). Qualitative research: rigour and qualitative research. *Bmj*, 311(6997), 109–112.

[110] McGrath, J. E. (1981). Dilemmatics: The study of research choices and dilemmas. *American Behavioral Scientist*, 25(2), 179–210.

[111] Merilinna, J. & Matinlassi, M. (2005). 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*.

[112] Miles, M. B., Huberman, A. M., Huberman, M. A., & Huberman, M. (1994). *Qualitative data analysis: An expanded sourcebook*. Sage Publications.

[113] Morgan, L. & Finnegan, P. (2010). Open innovation in secondary software firms: an exploration of managers' perceptions of open source software. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, 41(1), 76–95.

[114] Morse, J. M., Barrett, M., Mayan, M., Olson, K., & Spiers, J. (2002). Verification strategies for establishing reliability and validity in qualitative research. *International Journal of Qualitative Methods*, 1(2), 13–22.

[115] Munga, N., Fogwill, T., & Williams, Q. (2009). The adoption of open source software in business models: a red hat and ibm case study. In *Proceedings of the Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists* (pp. 112–121).: ACM.

[116] Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., & Ye, Y. (2002). Evolution patterns of open-source software systems and communities. In *Proceedings of the International Workshop on Principles of Software Evolution* (pp. 76–85).: ACM.

[117] O'Mahony, S. (2007). The governance of open source initiatives: what does it mean to be community managed? *Journal of Management & Governance*, 11(2), 139–150.

[118] O'Mahony, S. & Ferraro, F. (2007). The emergence of governance in an open source community. *Academy of Management Journal*, 50(5), 1079–1106.

[119] O'Reilly, M. & Parker, N. (2013). 'unsatisfactory saturation': a critical exploration of the notion of saturated sample sizes in qualitative research. *Qualitative Research*, 13(2), 190–197.

[120] Pearson, H. E. (2000). Open source licences: Open source—the death of proprietary systems? *Computer Law & Security Review*, 16(3), 151–156.

[121] Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3), 45–77.

[122] Peters, S. (2010). Best practices for creating an open source policy.

[123] Popp, K. M. (2015). *Best Practices for commercial use of open source software: Business models, processes and tools for managing open source software*. BoD–Books on Demand.

[124] Poslad, S., Buckle, P., & Hadingham, R. (2000). The fipa-os agent platform: Open source for open standards. In *Proceedings of the 5th International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agents*, volume 355.

[125] Riehle, D. (2007). The economic motivation of open source software: Stakeholder perspectives. *Computer*, 40(4), 25–32.

[126] Riehle, D. (2009). The commercial open source business model. In *SIGeBIZ Track of the Americas' Conference on Information Systems* (pp. 18–30).: Springer.

[127] Riehle, D. (2011). Lessons learned from using design patterns in industry projects. In *Transactions on Pattern Languages of Programming II* (pp. 1–15). Springer.

[128] Riehle, D. & Harutyunyan, N. (2017). Legal aspects – open source license compliance in software supply chains. In B. Fitzgerald, A. Mockus, & M. Zhou (Eds.), *No.099 Towards Engineering Free/Libre Open Source Software (FLOSS) Ecosystems for Impact and Sustainability*: NII Shonan Meeting.

[129] Riehle, D. & Lempetzeder, B. (2014). *Erfolgsmethoden der Open-Source-Governance und-Compliance*. Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU).

[130] Ritchie, J., Lewis, J., Nicholls, C. M., Ormston, R., et al. (2013). *Qualitative research practice: A guide for social science students and researchers*. Sage Publications.

[131] Ruffin, C. & Ebert, C. (2004). Using open source software in product development: A primer. *IEEE Software*, 21(1), 82–86.

[132] Runeson, P. & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2).

[133] Russell, C. K. & Gregory, D. M. (2003). Evaluation of qualitative research studies. *Evidence-Based Nursing*, 6(2), 36–40.

[134] Saini, S. K., Krishnan, C., & Rajaram, L. (2010). open source adoption index: quantifying foss adoption by an organisation. *International Journal of Open Source Software and Processes (IJOSSP)*, 2(3), 48–60.

[135] Schöttle, H. & Steger, U. (2015). Managing open source software in the corporate environment. *Computer Law Review International*, 16(1), 1–7.

[136] Schreiber, A. & Haupt, C. (2017). Sharing knowledge about open source licenses at dlr. In *Proceedings of the 13th International Symposium on Open Collaboration*: ACM.

[137] Shaikh, M. & Cornford, T. (2009). Innovating with open-sourcing: governance concerns for managers. In *Proceedings of the 15th Americas' Conference on Information Systems*.

[138] Shenton, A. K. (2004). Strategies for ensuring trustworthiness in qualitative research projects. *Education for Information*, 22(2), 63–75.

[139] Stam, W. (2009). When does community participation enhance the performance of open source software companies? *Research Policy*, 38(8), 1288–1299.

[140] Stewart, K., Odence, P., & Rockett, E. (2010). Software package data exchange (spdx) specification. *IFOSS L. Rev.*, 2.

[141] Stol, K.-J. & Ali Babar, M. (2010). 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.

[142] Tapia, L. M., López, L., Ayala, C. P., & Annosi, M. C. (2015). Towards an oss adoption business impact assessment. In *IFIP Working Conference on The Practice of Enterprise Modeling* (pp. 289–305).: Springer.

[143] Ternier, S., Verbert, K., Parra, G., Vandeputte, B., Klerkx, J., Duval, E., Ordonez, V., & Ochoa, X. (2009). The ariadne infrastructure for managing and storing metadata. *IEEE Internet Computing*, 13(4), 18–25.

[144] Torkar, R., Minoves, P., & Garrigós, J. (2011). Adopting free/libre/open source software practices, techniques and methods for industrial use. *Journal of the Association for Information Systems*, 12(1).

[145] Trochim, W. M. (1989). Outcome pattern matching and program theory. *Evaluation and Program Planning*, 12(4), 355–366.

[146] Trochim, W. M. (2006). Qualitative measures. *Research Measures Knowledge Base*, 361, 29–31.

[147] Umarji, M., Sim, S. E., & Lopes, C. (2008). Archetypal internet-scale source code searching. In *IFIP International Conference on Open Source Systems* (pp. 257–263).: Springer.

[148] Wang, H. & Wang, C. (2001). Open source software adoption: A status report. *IEEE Software*, 18(2), 90–95.

[149] Webster, J. & Watson, R. T. (2002). Analyzing the past to prepare for the future: Writing a literature review. *MIS Quarterly*, (pp. xiii–xxiii).

[150] Weiss, M. (2010). Profiting from open source. In *Proceedings of the 15th European Conference on Pattern Languages of Programs*, EuroPLoP '10 (pp. 5:1–5:8).: ACM.

[151] Weiss, M. (2018). Business of open source: A case study of integrating existing patterns through narratives. In *Proceedings of the 23rd European Conference on Pattern Languages of Programs*, EuroPLoP '18 (pp. 23:1–23:4).: ACM.

[152] West, J. (2003). How open is open enough?: Melding proprietary and open source platform strategies. *Research Policy*, 32(7), 1259–1285.

[153] West, J. et al. (2007). The economic realities of open standards: Black, white and many shades of gray. *Standards and Public Policy*, 87.

[154] West, J. & Gallagher, S. (2004). Key challenges of open innovation: lessons from open source software. *San Jose State College of Business, mimeo*.

[155] von Willebrand, M. & Patanen, M.-P. (2010). Package review as a part of free and open source software compliance. *IFOSS L. Rev.*, 2.

[156] Wolff-Marting, V., Hannebauer, C., & Gruhn, V. (2013). Patterns for tearing down contribution barriers to floss projects. In *2013 IEEE 12th International Conference on Intelligent Software Methodologies, Tools and Techniques (SoMeT)* (pp. 9–14).: IEEE.

[157] Yin, R. K. (2017). *Case study research and applications: Design and Methods*. Sage Publications.

[158] Zhu, S. (2007). Patent rights under foss licensing schemes. *Shidler Journal of Law, Commerce & Technology*, 4(1).

[159] Zimmermann, J.-B. & Jullien, N. (2007). Free/libre/open source software: lessons for intellectual property rights management in a knowledge-based economy. *The Icfai Journal of Cyber Law*, 6(3), 19–36.

This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Egenolff-Berner Garamond, a revival of Claude Garamont's humanist typeface. The above illustration, "Science Experiment 02", was created by Ben Schlitter and released under CC BY-NC-ND 3.0. A template that can be used to format a PhD thesis with this look and feel has been released under the permissive MIT (XII) license, and can be found online at github.com/suchow/Dissertate or from its author, Jordan Suchow, at suchow@post.harvard.edu. This template was adapted for Friedrich-Alexander-University Erlangen-Nuremberg (FAU) by Nikolay Harutyunyan. You can get the source code to this template on Overleaf or from him, at nikolay.harutyunyan [at] fau.de.