Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik

Viktoriya Promyshlyanska

MASTER THESIS

Theory of Industry Best Practices of FLOSS Governance and Compliance

Submitted on 13. June 2017

Supervisor: Prof. Dr. Dirk Riehle, M.B.A.
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander University Erlangen-Nürnberg

Versicherung

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Viktoriya Promyshlyanska

____

Nuremberg, 13. June 2017

License

This work is licensed under the Creative Commons Attribution 4.0 International license (CC BY 4.0), see https://creativecommons.org/licenses/by/4.0/

Viktoriya Promyshlyanska

____

Nuremberg, 13. June 2017

Abstract

Using open source code in commercial software development is gaining more and more momentum during the recent years. While providing benefits along all three dimensions of the software development's magic triangle of cost, time and scope, use of open source in commercial setting holds certain risks and challenges that can be addressed with FLOSS governance and compliance in the organization. However, the surveys show that about a half of the companies dealing with open source in their software development do not have any specific FLOSS governance procedures. The objective of this thesis is to develop a theory of FLOSS governance and compliance best practices in software development companies. The study uses case study research methodology applied to five German companies. The best practices are derived from the data collected during semi-structured interviews with the help of Qualitative Data Analysis. Key research findings are summarized and the full list of derived best practices in the form of best practice patterns is presented. The formulated best practices in the categories "Policies", "Processes" and "People and Tools" can be used by software companies to leverage the advantages of using open source while mitigating the associated risks.

Keywords

Open source software, FLOSS, FOSS, proprietary software, FLOSS government, open source license compliance, software development management, case study, best practice

Contents

1 Introduction

1.1 Original Thesis Goals

The goal of this master thesis was to build a theory of the Best Industry Practices of FLOSS Governance and Compliance through exploratory multiple case studies conducted with five specifically selected companies.

The following work packages were planned to be completed in frames of this thesis:

- Define the research question and research approach
- Conduct the literature review following the methodology of Webster & Watson (2002)
- Review and adjust the initial research question
- Prepare and conduct the first case study research
- Analyze collected data following Qualitative Data Analysis methodology
- Review and adjust the initial case study plan
- Conduct the remaining four case studies
- Analyze newly collected data following Qualitative Data Analysis methodology
- Aggregate the results of the case studies' analysis to determine the common best practices
- Summarize the most important research findings

The focus of this thesis is set on industrial best practices of FLOSS governance and compliance, therefore the case study research methodology was chosen. The goal was to compare the best practices suggested by considerably limited scientific literature available on the topic and the common best practices used in the industry. Since there is no solid theory of FLOSS governance and compliance best practices to be found in the literature, this thesis was called to make a contribution to this novel research area.

In order to achieve optimal aggregation of the literature and industrial insights, literature review was selected to be one of the first steps of the master thesis. The goal of the literature review was to gain an overview of the available research, refine the research question, define the top-level categories of best practices and prepare for the future case studies.

The companies for the case studies were to be selected through theoretical sampling based on the various characteristics, such as: type of business, size, maturity.

Case studies were to be carried out following the methodology suggested by Yin (2013). The goal was to build up an iterative process by adjusting and improving the case study plan based on the results of each conducted case study.

The research results presentation was called to provide a clear summary of the most important research findings, while also providing the handbook of concisely formulated best practices derived from the research.

1.2 Changes to Thesis Goals

Due to the time constraints and limited availability of the industry partners, it was not possible to conduct two interviews with Company 3, following the initial plan. Only one interview with this partner was conducted in frames of this thesis.

2 Research Chapter

2.1 Introduction

The beginning of Open Source era was marked with the establishment of GNU project (http://www.gnu.org/) and the Free Software Foundation (http://www.fsf.org/) in 1980's. Free Software is defined by GNU as

"Software that respects users' freedom and community. Roughly, it means that the users have the freedom to run, copy, distribute, study, change and improve the software. Thus, "free software" is a matter of liberty, not price. To understand the concept, you should think of "free" as in "free speech," not as in "free beer". We sometimes call it "libre software," borrowing the French or Spanish word for "free" as in freedom, to show we do not mean the software is gratis." ("What is Free Software?," n.d.)

The term "Open source software" defines almost the same software category, with the major difference being that free software is a social movement, while open source is a development methodology ("Free Software Philosophy," n.d.),(Stewart & Gosain, 2006). The differences in these definitions are not relevant for the conducted research, therefore terms "Open Source", "OSS", "FOSS" and "FLOSS" will be used as synonyms in this thesis.

Using open source software is a popular code reuse method in industrial software development. (Chang, Lee, & Yi, 2010). 90% of respondents of "Future of Open Source Survey" conducted in 2016 by North Bridge and Black Duck (Hammond, Santinelli, Billings, & Ledingham, 2016) stated that Open Source improves efficiency, interoperability and innovation. The reasons why companies use FLOSS include possibility to speed up the development process and focus own resources on core competence; no licensing costs; quality of solutions; freedom from vendor lock-in; easy customization. (Hammond et al., 2016), (BearingPoint, 2012). According to Jeff Hammond, principal analyst at Forrester Research, open source is a "silver bullet" that allows simultaneous improvement along all three dimensions of the software "iron triangle" of cost, schedule and features. (Black Duck & BearingPoint, 2013)

With 57 million repositories currently available on GitHub (Firestine, 2017) – the biggest open source code host in the world – and tempting benefits that open source reuse provides, using open source components in commercial product development has become a natural practice. "We use open source software in nearly everything we do because it helps us produce higher quality software, better and faster." – states Dr. Yunjae Jung, Principal Specialist at Samsung SDS (Black Duck, 2015). However, according to Hammond et al. (2016), 50% of companies have no formal policy for selecting and approving open source code.

While using FLOSS in software product development can be highly beneficial, it is necessary to be aware of the associated risks and potential problems (Fendt, Jaeger, & Serrano, 2016). Some of the most significant problems include:

- license violations associated with legal implications that can damage the brand and lead to significant material losses (Chang et al., 2010)
- a wide range of available OSS components making the selection process challenging and the importance of thorough evaluation (Höst, Oručević-Alagić, & Runeson, 2011)
- potential technical failure and security risks (BearingPoint, 2012)

Satisfying license obligations is the basic responsibility of every company exploiting FOSS in their products. However, with more than 360 different open source licenses existing nowadays

(Fendt et al., 2016), this is not a trivial task. In order to be able to mitigate the risks and leverage the benefits of using open source to its fullest, it is necessary to establish effective FLOSS governance in the company (Höst et al., 2011).

The goal of this research is to formulate the best practices of FLOSS governance and compliance suggested both by scientific literature and by industrial experience, which can be used as guidelines by companies on their journey to effective FLOSS governance.

The paper employs exploratory case study research methodology, while conducting case studies at five different companies. Collected data is analyzed using qualitative data analysis tools in order to derive the best practices.

Following the introduction, key findings from conducted literature review are presented in section 2.2 of this paper. In section 2.3, the research question is outlined. Section 2.4 describes the research approach, while section 2.5 outlines used data sources and provides company context for the participating industry partners. In section 2.6, the main findings from the case studies are presented, while section 2.7 briefly discusses the study's limitations and outlook. Elaboration chapter contains full results of the conducted literature review under section 3.1 as well as the entire list of derived best practices in form of a concise practices handbook under 3.2.

2.2 Related Work

With the rise of open source software in the recent decades, there is lots of scientific research available on different aspects of FOSS phenomena, such as open source development philosophy, motivation behind participating in FOSS projects, comparison of open source and proprietary software products and so on. Most of this research is not relevant for the purposes of this thesis. In frames of this research work, it was a clear purpose to investigate in depth solely the works, focusing exactly on the area of interest, which is: recommendable practices for FOSS governance and license compliance in commercial setting. I.e. how companies can deal with the specifics of reusing open source code / components in the development of their (commercial) products most efficiently.

The literature review was performed following the methodology suggested by Webster & Watson (2002). Twenty papers were selected for literature review through literature sampling. In order to get the broader and more practice related view on the topic, whitepapers of prominent companies successful in the field of FOSS governance were taken into consideration next to scientific publications. Based on the literature review, certain categories and sub-categories on the topic were gradually determined. The higher-level categories include "People", "Processes", "Tools", "Policies" and "Strategy". These categories are in line with previous research. R. Kemp (Kemp, 2010) mentions that FLOSS governance is a complex task and can be viewed as consisting of several building blocks, such as: the people context, the strategic context, the policy context and the process context. He also distinguishes an "achievements to date" context, which was left out of scope of this research. Each of the categories contains multiple sub-categories, which will be discussed further. The determined categories composed a comprehensive literature review framework, which was filled out for each of the reviewed papers. The most important findings are summarized further in this chapter and the complete results table can be found in appendix.

The literature suggests that the first step on the way to successful FOSS governance is the definition of company's FOSS policy (Haddad, 2010d), (Mutkoski, 2004), (Kemp, 2010), (BearingPoint, 2012). Since we chose not to include the “initial establishment of FLOSS governance” and “strategy” dimensions into the research scope, the results of conducted case studies concentrate on the specific practices regarding FLOSS policies in the company, and not on the high-level FOSS policy concept.

One of the important points ensured by company’s FOSS policy can be creating and maintaining good relationship with FOSS community, since the community provides source code, support, documentation and testing for the FOSS software (Haddad, 2010d). Many companies have their employees as members of FOSS communities and this proves to be a good practice (Chang et al., 2010). The practice of getting involved in FOSS communities was repeatedly mentioned in the interviews with industry partners as a highly recommendable proceeding, fostering the communication with FOSS communities and contributing to the FOSS components’ selection process.

Contributing changes back to the open source project can be beneficial, since it makes it easier to include updates / bug fixes of the open source component (Höst et al., 2011). However, this might cause leaking of essential organizational IP (Höst et al., 2011). Most of the industry partners emphasize the importance of contributing back and mention that usually its benefits outweigh the risks connected with organizational IP and other legal issues.

The majority of the reviewed papers emphasize the importance of trainings. Haddad (2011b) and Koltun (2011) point out that all relevant stakeholders, i.e. all staff that come into contact with FOSS either on procurement, development or shipping stage, need to be sufficiently educated on FLOSS policies. Case study results confirm these findings. Each partner company either has open source trainings in place or realizes the need for them and plans to introduce such trainings in the nearest future.

Another mechanism fostering FLOSS governance and compliance is providing clear practical guidelines covering, for example, the description of most commonly used FOSS licenses (Haddad, 2010b), whitelists and blacklists of approved/denied FOSS packages and licenses (BearingPoint, 2012), common mistakes and how to avoid them (Haddad, 2010c). Conducted case studies are in line with these recommendations. Most of the industry partners involved either already have such guidelines developed or are currently working on them, depending on the company’s maturity level.

Carrying out FLOSS governance, it is necessary to remember that there are two entry points for FOSS into a product: developer and supplier (BearingPoint, 2012). While the practices outlined above provide guidance on how to manage the “developer” entry point, it is crucial to realize that similar practices must be applied to third party supplied software, since the distributor is responsible for all the code contained in the product. It is important to adjust supply chain procedures to assure FOSS license compliance of acquired components (Haddad, 2010c). A best practice is to request detail FOSS usage declarations from the binary components suppliers with information about integrated open source components and the measures taken to fulfill the license requirements (Fendt et al., 2016). Another best practice mentioned by our industry partners is clearly regulating the responsibilities for the code with the help of an explicit contract with the supplier.

Defined FOSS policies should be implemented in form of processes, which have to be efficient and light weight (Haddad, 2010b). The FLOSS processes have to take the strain of FLOSS governance (Kemp, 2010). It is crucial that the overhead of the process is outweighed by the benefits that the process yields. The process descriptions found in (Haddad, 2010c), (Haddad, 2011a), (Koltun, 2011), (Fendt et al., 2016), (Black Duck, 2016a), (Black Duck & BearingPoint, 2013) were analyzed and aggregated into a sample FLOSS governance process, which can be found in section 3.1, with detailed description of each phase and recommendations for its successful implementation.

In order to be effectively implemented the process dimension must be supported by “People” and “Tools”. According to Kemp (2010), in order to tame a company’s open source usage and ensure effective FLOSS governance, a number of stakeholder groups should be involved, while integrating and cooperating in a supportive and positive way. The majority of reviewed papers suggest the need to establish a core and extended FLOSS governance team, as well as to assign a FLOSS Compliance Manager (also known as Compliance Officer, Director of Open Source, etc.) - the head of the core FLOSS governance team. Conducted research goes in line with these findings from the literature. The importance of the core and extended team as well as an Open Source Director position was repeatedly brought up by our industry partners during the interviews.

Core team, called Open Source Review Board (OSRB) in (Haddad, 2010c), should consist of Legal Representative(s), Engineering and Product Team Representative(s) and Compliance Officer. Similar structure of the centralized FLOSS governance team was mentioned in the Interview 3. The OSRB reviews and approves FLOSS usage requests and serves as steering committee for company’s FLOSS strategy (Haddad, 2010d). According to the Interview 3, Open Source Director also plays an important role in strategic open source foresight. From strategic point of view, it is crucial to gain executive C-level commitment to FLOSS governance and compliance (Haddad, 2010b), (Kemp, 2010), (Black Duck, 2014). The extended team should be built of various individuals from different departments, who contribute to FLOSS governance effort on an on-demand basis, according to the tasks assignments from the core team (Haddad, 2010c). These recommendations are confirmed with our research

According to BearingPoint (BearingPoint, 2012), it is impossible to take a firm grasp on FOSS governance with manual processes. The complexity of the associated tasks is too sheer, making manual methods doomed for incompleteness, inconsistency and non-sufficient speed to keep up with the modern agile development lifecycle. Therefore, there is a strong need for automation (Haddad, 2011b), (BearingPoint, 2012), (Black Duck, 2016a). The absolute majority of the reviewed papers emphasize the importance of using source code-scanning tools. The case studies results go in line with these findings.

Other advised tools mentioned in the literature include project management tools, Bill of materials (BoM) difference tool; dependency tracking tool; binary scanning tool; code janitor tool; source code peer review tool; license to requirements mapper; product FOSS documentation generator; product FOSS code collector (Haddad, 2010c), (Fendt et al., 2016). However, none of these tools were brought up during case studies.

The most recent papers mention holistic solutions, uniting all of the tools mentioned above and enabling companies to streamline the entire FOSS process. One example of such tool is Black Duck software suit, which automates search, selection, approval, validation and monitoring of FLOSS in the product development. It enables companies to integrate FLOSS governance tasks into software development process in an effective way, while avoiding unnecessary delays and developers‘ resistance to the process overhead (Black Duck, 2016a). Whitepapers (Black Duck, 2015) and (Black Duck, 2014) provide the success stories of the Black Duck customers who have managed to take a grasp on their FLOSS governance with the help of the process automation. Such holistic solutions are not currently used by our industry partners, not even by those in the highest maturity level.

In section 3.1 the full literature review results are presented describing the current state of the research in the field as well as all the practices found in the literature in greater detail.

2.3 Research Question

The central research question of this thesis is:

“What are the industry best practices of FLOSS Governance and Compliance?”

During the literature review the following categories of the best practices arose, which were consequently confirmed by the results of qualitative data analysis of the data collected during the case studies:

- People & Tools
- Policies
- Processes

Other categories suggested by the literature review, such as “Strategy” and “Initial FLOSS governance establishment” were intentionally left out of scope of this thesis.

The goal of the thesis is to derive a theory of best practices in the above mentioned categories and present them in form of a practical handbook of best practice patterns as well as to concisely summarize the most important research findings.

2.4 Research Approach

After elicitating the categories of the best practices during the literature review, case study research methodology was selected for deriving the best industrial practices in these categories. This research method was chosen since the research question concerns itself with a complex real life phenomenon – industrial FLOSS governance. The elicitation of the best practices required a comprehensive and “in-depths” investigation of this phenomenon in its real-life context. This thesis follows the methodology suggested by Yin (2013) and takes the paper of Harutyunyan (2016) for reference regarding practical application of the Yin’s methodology. Yin defines the following stages of the case study: preparing for the case studies, selecting the candidates for the case studies, collecting and analyzing data, deriving and presenting results.

In the first step, the embedded multiple-case design was selected for the case study, with best practices of FLOSS governance in the organization being an embedded unit of analysis, and each company being a separate case. Preparation phase also included developing the case study protocol, which can be found in the Appendix A. The used template for case study protocol is suggested by Brereton et al. (2008). Checklist for Software Engineering Case Study Research (Höst & Runeson, 2007) was filled in for the sake of quality assurance of the conducted research. It can be found in Appendix C.

In order to contribute to the versatility and objectivity of the derived results, screening of available case candidates was performed for the selection of industrial partners. The cases were selected based on the following criteria:

- Type of product / service
- Type of customer
- Market position
- Size
- Maturity

The goal was to investigate different types of companies in order to be able to comprehend the differences in best practices depending on the characteristics of the company.

Following Yin's recommendations, an iterative approach was used during this research work, i.e. interview questions, case study design and codebook were further improved after finishing each case study based on the recent learnings. The data collection was realized mostly through interviews with industry partners. Used data sources are described in detail in the next chapter. Data analysis using QDAcity – a qualitative data analysis tool developed in Open Source Research Group at FAU – was performed in parallel with data collection.

Based on the results of qualitative data analysis of the data collected in all five cases, a handbook of best practices of FLOSS governance and compliance was developed. The practices are presented in the form of patterns describing the actor, context, problem and solution in the Elaboration chapter, in section 3.2. The " Research Results" part of the research chapter provides a summary of the most important findings of the conducted case study and presents an overview and interconnections between the derived best practices.

2.5 Used Data Sources

The main data source for this thesis results were semi-structured interviews conducted with the selected industrial partners. Data gathering was performed in close cooperation with Nikolay Haratyunyan. The following table provides an overview of the selected companies:

Size \ Maturity level	Startup	Growth	Mature
Small			Company 5
Medium		Company 2 Company 4	Company 3
Large		Company 1	

Table 1. Classification of industry partners

The companies on "Startup" maturity level were intentionally left out, since the goal of this research was to derive the best practices, which are unlikely to be solidly formed on the "Startup" level of maturity. Out of the five participating companies, two create proprietary software products, two create open source software products and one is specialized in consulting on open source topics.

The following data was collected:

- Two interviews with Company 1
- One interview with Company 2
- One interview with Company 3
- One interview with Company 4
- One interview with Company 5

Company 1 is a large IT services provider that mostly uses project oriented approach. They use open source components in their software development and are currently in the "Growth" maturity level. Two interviews were conducted with this company: the first one with the head of functionality of the development team and the project manager of one of the company's biggest projects; the second one -with the head of requirement engineering and solution design team and the project manager of another one of the company's biggest projects.

Company 2 is an experienced medium-sized open source consulting company. This company specializes in supporting their clients in using open source components. This includes both software development and consulting activities. The interview was conducted with the company's CEO and provided valuable insight into best practices of FLOSS usage and governance from a somewhat different perspective. For instance, this industry partner provided us with a number of examples of negative consequences caused by insufficient FLOSS governance.

Company 3 is a medium-sized mature open source software company. The interview was conducted with the company's Director of Open Source. This interview provided an exceptional insight in the FLOSS governance team organization and particularly the role and responsibilities of the Director of Open source in a company.

Company 4 is a fast growing medium-sized software product development company. The company was originally specialized in open source software development, but has been partly shifting to the commercial business models in the recent years. The interview with the company's head research and innovation engineer let us gain insight into the practices adopted by companies finding themselves in the fast growth phase, where the need for well-established FLOSS governance processes is constantly increasing, while not everything is settled yet.

Company 5 is an IT department of one of the biggest German car manufacturers. The interview was conducted with the Open Source Compliance Manager and provided a lot of highly valuable information on FLOSS governance and compliance practices used by this mature company in their software development.

2.6 Research Results

As a result of this research work, a theory of best industry practices of FLOSS governance and compliance was formulated by presenting the best practices in the form of patterns in the following categories:

- Policies
- Processes
- People and Tools

The full list of best practices can be found in the elaboration chapter. In this section, the results are summarized and most important research findings are presented.

Figure 1 illustrates the way different best practice categories interconnect with each other. "People" and "Tools" are brought together under the common category "Infrastructure", which realizes FLOSS governance processes, defined by FLOSS governance policies.

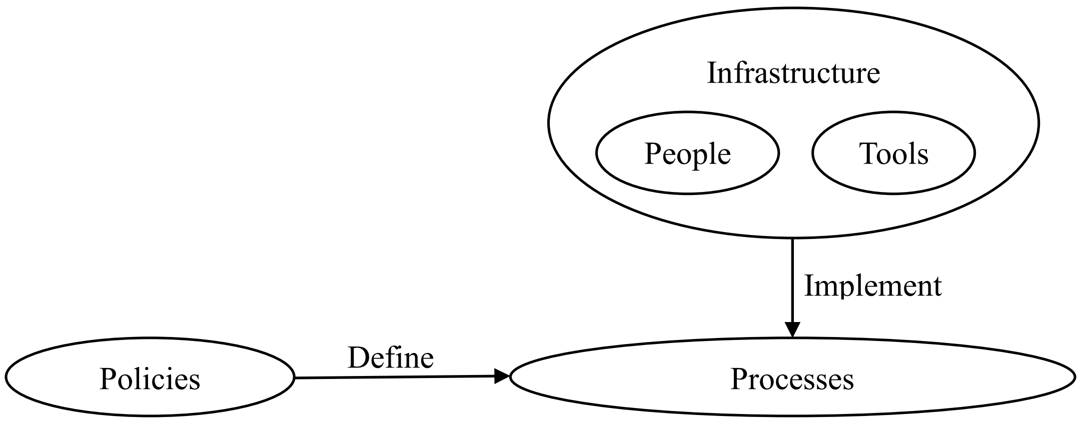

Figure 1. Relations between categories of best practices

A total number of thirteen best practices were derived during conducted case study research:

- six practices in the category “Policies” describing advantageous approaches which company’s FLOSS policy should embrace;
- three practices in the “Processes” category outlining crucial phases of the FLOSS governance process and recommendations for their successful implementation;
- four practices in the category “People and tools” providing insights into advisable staffing and team structure for successful FLOSS governance, as well as recommendations for tool usage.

Figure 2 presents the interconnections between all the derived best practices. Different shape colors stand for different categories:

- blue – policies,
- yellow – processes,
- orange – people,
- green – tools.

Different types of connecting lines indicate various types of relations, which are explained in the figure’s legend. The pointed end of the lines implies the direction of relation.

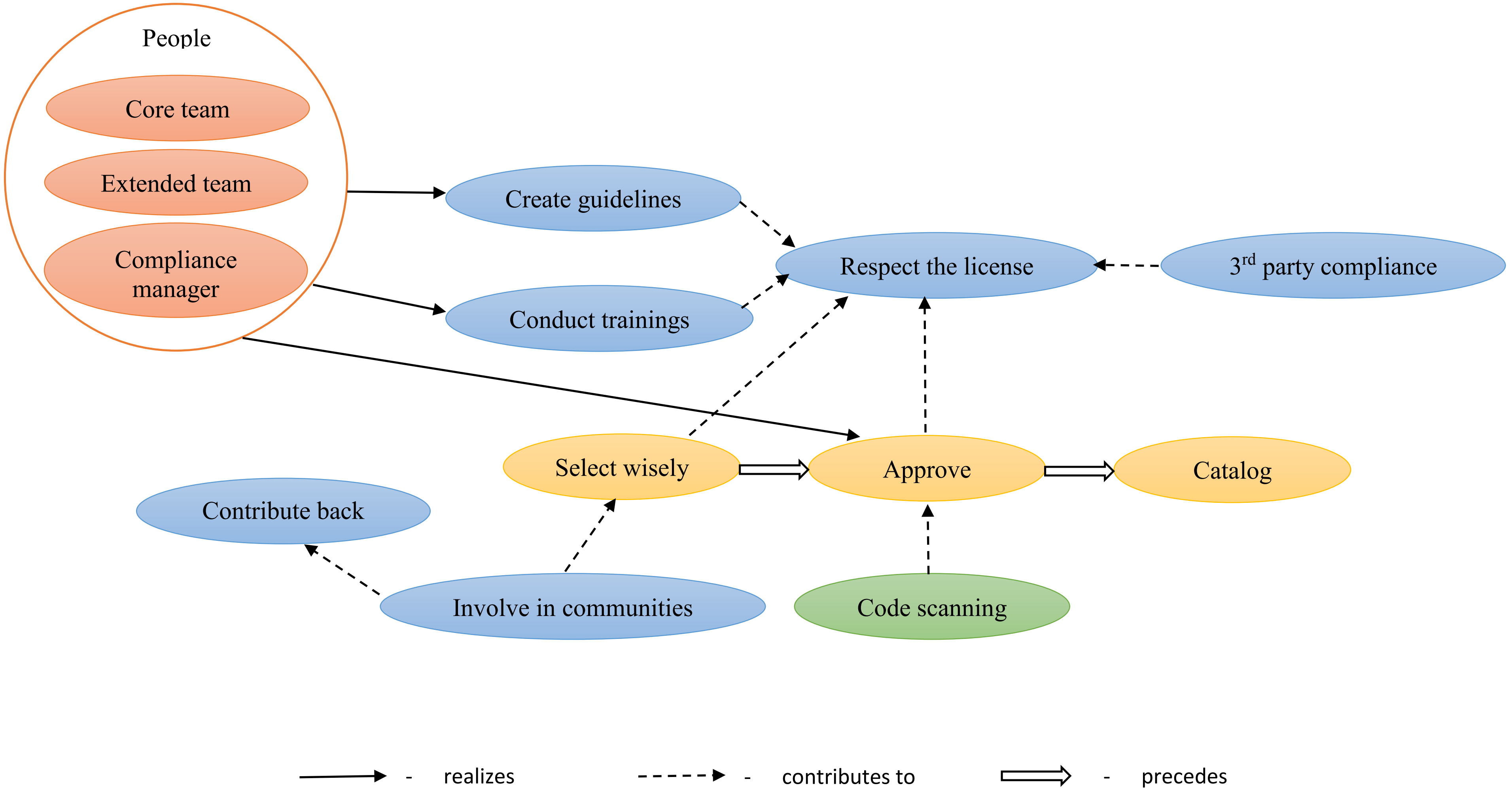

Figure 2. Overview of derived best practices

In the elaboration chapter derived practices are presented in the form of best practice patterns, highlighting the actor executing the practice, the context in which the practice should be used, the problem it solves and the solution it provides. Each practice has references to the interviews and literature sources where it was mentioned. As illustrated by figure 2, derived practices are interconnected. These interconnections are mentioned in the best practice patterns. Table 2 provides an example of how a best practice description looks like:

Name	Respect the license
Actor	Every employee
Context	You are using / planning to use open source components in your software development
Problem	How to ensure compliance of your software?
Solution	Be aware of the different open source licenses and ensure compliant usage of open source in software development. Use *Create compliance guidelines* and *Conduct trainings* practices to provide preventive measures. Include licensing aspect into selection criteria in frames of *Select wisely* practice. Apply *Approve* and *Using code scanning tools* practices to ensure that only approved open source software is used in the products and thus assure software compliance. Although, you should not rely solely on scanning tool, but groom the understanding and respect to open source licenses in your employees instead. In case legal department concludes that a certain license cannot be used in a certain software piece – an alternative solution must be selected. You should try to detect any incompliances as early in the development process as possible. However, even if detected in the late stage – the incompliances must be resolved before software distribution. Conduct regular software audits to foster license compliance. You should keep track of all the used open source components, their licenses and corresponding obligations. After software distribution, the license obligations must be fulfilled.
References	Interviews: 3, 5; Literature: every reviewed paper

Table 2. Example of a best practice pattern

Respect the license practice presented in the example is one of the central practices in the "Policies" category, since FLOSS License Compliance is one of the main goals of FLOSS governance (Kemp, 2010). Another aspect that can come up in this context is the problem of license incompatibility. Sometimes license incompatibility can be solved with separate compilation of different parts of the software. An example of this practice provides Company 4, who release the base version of their software under GPL and develop separately compiled commercially licensed plugins for it. This trick can be used by companies tackling the problem of license incompatibility.

Contributing to successful implementation of Respect the license practice are Creating guidelines and Conducting trainings. Based on the conducted case studies we can see that companies in the Growth maturity level (Company 1, 2 and 4) are not conducting such trainings for their employees at the moment. Whereby for the medium-sized companies 2 and 4 this is caused by company's reliance on the internal expertise of the employees and well-established informal connections between staff members; while the large-sized Company 1 is adjusting their training plans at the moment, which is typical for a company gaining maturity. Our partner from Company 1 emphasizes the importance of trainings by his statement: "such trainings make your mind reset for compliance issues". Both of the mature companies participating in the case study conduct FLOSS governance and compliance trainings for their employees. Company 3 integrates compliance training in the regular project onboarding activities. Company 5 employs a self-developed web-based compliance training. Research findings suggest that software development companies striving for successful FLOSS governance and compliance should conduct corresponding trainings for their employees in any of suggested formats.

Another important practice, mentioned by all our industry partners is Contribute back. Our partner from Company 2 emphasizes the importance always to give the code modifications, either bug fixes or new features, back to community. They provide examples of how not contributing back can result in severe problems, e.g. when a company creates a patch with new features for an open source component and does not contribute it back. In this case, the new version of the open source component cannot be employed by the company without extra effort, same as bug and security vulnerability fixes from the community cannot be simply adopted. These problems can be avoided by contributing all the performed changes back to community. "[taking into account] The advantages of open source and having it patched just for yourself - it doesn't give you any advantage" – states our partner in interview 2. Realizing the importance of this point, our partners from Company 3 include considerations on how easy it will be to contribute back to open source community into OSS component selection process. He says: "We prefer not to fork or keep our own branch and code. It is very costly, it is hard on the project, it is hard on the companies, it is hard on everybody involved. So the best practice is: we contribute everything back. That's why earlier we talked about one of my criteria is what's our ability to contribute to the project." Based on the research findings, I suggest that software companies should employ the practice of contributing changes back to the open source community.

Closely connected to Contribute back is the practice Involve in FOSS communities. Most of our industry partners mention that having employees as members of FOSS communities is beneficial both for Contribute back and Select wisely practices. It is easier to get information about community, communicate with community and give changes back to community when your employees are involved in those. In Interview 5, our partner emphasizes that it is important to pay attention to legal regulations and company IP while contributing to FOSS communities. It is necessary to decide whether employees should contribute as individuals or from the company's name, eventually get clear with CLAs, ensure that the contributed code is high quality and its release does not harm the company's core IP. Useful instruments for achieving these goals can be peer or legal reviews. Based on conducted research, a recommendation for software companies is to have their employees as members of FOSS communities, while keeping in mind associated legal and IP issues.

The practices in the "Processes" category concentrate on three crucial phases of the FLOSS governance process: selection, approval and cataloging. Select wisely practice highlights the importance of smart selection of open source components to use and provides recommended selection criteria employed by industry, such as: technological aspects, legal aspects and community aspects. It is recommended to opt for diverse communities not driven by a single company in order to avoid vendor lock-in. It is emphasized that being Involved in FOSS communities makes the assessment of communities much easier. Following this recommendations in their selection process, software companies can leverage the use of open source to its fullest.

Approval step in the FLOSS governance process is essential for achieving FLOSS compliance. Having open source use approved before implementing it allows companies to stay compliant. However, there is a need for balance between checking all necessary changes and turning approval into process bottleneck. Mature companies tend not to rely on the centralized team for approving every small change, but rather educate their engineers and only turn to the core team in case of bigger occurrences, such as an entirely new component to be used or a new unknown license coming up. Nevertheless, on the early stages of establishing FLOSS governance, it is recommendable to exercise approval as a centralized task. In the interview 3, our partner emphasizes that in case a request cannot be approved – it is crucial to communicate the reasons clearly to avoid conflicts and raise the open source awareness among software engineers. “I think the best practice there is just ensure that the people making the request understand why the request can not be approved. If you clearly communicate what the issue is then they can come up with alternatives on how to mitigate that,” – states our partner.

Cataloging adopted open source components after their use is critical in order to keep up a clear picture of what is inside the company’s software. A negative example brought up by our partners from Company 2 pictures a banking company that had seven different versions of the same open source library in their system, which eventually led to the system collapse due to two of the versions being incompatible. This situation could have been avoided by proper cataloging of the software components when bringing them into the system. Thus, a recommendation for software companies using FLOSS is to thoroughly document its use in order to avoid unexpected problems and foster the component reuse.

Practices presented under “People and Tools” category indicate the recommendable team structure for FLOSS governance team and their required competences. Similar to the recommendations found in the literature, industrial practices suggest assigning a Director of Open Source, a centralized FLOSS governance team, and an extended team. From the conducted case studies we can see that while mature companies have the teams and responsibilities for FLOSS governance clearly established, the companies finding themselves in growth phase, might not have them well-established yet. For example, medium-sized Company 2 does not have a centralized FLOSS governance team. Being a specialized open source consultancy in the growing phase, they can rely on the deep open source expertise of their employees and omit the centralized governance unit. However, their CEO mentions that “if you have thousands of technical people, you cannot have everyone make their own decisions. Then you have so many different tools, it is going to get tricky. So, you have to find some way of doing the centralized decision. It can be a board or one person, but you have to find some way of making those decisions.” This is confirmed by another partner in Interview 3, claiming: “I think companies not doing an open source review board are making a mistake.” Based on these findings it can be concluded that software companies should have a centralized team dealing with open source, especially in case if the company is large and / or the FLOSS expertise of its employees is limited.

In contrast to a big number of various tools that can facilitate FLOSS governance mentioned in the literature, our industry partners only emphasize the importance of code scanners. Whereby the divergence between mature and growing companies can be noticed. While mature companies are performing code scanning automatically on each software build (Company 3), growing companies are still on their way towards establishing code scanning procedures (Company 1, 4). The research findings suggest that companies dealing with open source should use code scanning tool. Although one of our industry partners emphasizes, that while automated code scans are highly beneficial, they can not replace raising awareness and encouraging responsible FOSS use among company’s employees.

Although other tools, such as software inventories or SPDX were also sporadically mentioned, these references were not considered enough for forming the best practice patterns in frames of this study. Based on conducted research, we can conclude that SPDX is an emerging standard that most of the companies have heard of and believe it is worth using, but only a few are already actually using it. Among our partners, only Company 5 is using SPDX in their work. They require an SPDX document with each software delivery from their suppliers. In Interview 3, our partner mentions: "It [SPDX] is coming, but it has not been universally adapted yet." Based on these findings, it makes sense for software companies to look into SPDX standard and incorporate it in FLOSS governance procedures, for example, by asking 3rd party software suppliers to submit an SPDX document together with software delivery.

In this section, the overview of best practices of FLOSS governance and compliance derived in frames of this master thesis was presented. Selected practices were elaborated in detail, supporting pieces of evidence from the interview raw data were presented and the implications for software companies were outlined. Elaboration chapter (Section 3.2) provides the full list of derived best practices in form of best practice patterns containing practical recommendations for software companies.

2.7 Limitations and Conclusions

Main limitations of this study include the restricted number of investigated cases and the limited sources of case study evidence. In most of the cases, it was only possible to conduct one interview due to time restrictions and limited availability of the industry partners. Collecting other sources of evidence, such as documentation (e.g. company's compliance guidelines, training materials) or other artifacts was not possible, since most of the relevant artifacts are strictly confidential.

Conducted research provides a handbook of thirteen best practices of FLOSS governance and compliance derived from industrial experience of five different companies. The identified best practices are a partial theory and can be completed with additional practices in the identified categories as well as beyond the scope of these categories. Suggested theory of best practices could be a subject of future confirmatory research. The research findings mostly go in line with the results of the conducted literature review, while contradicting some of the points mentioned in literature and adding industrial insights and practical recommendations on practices' implementation. Presented practices can be used by companies striving for successful open source governance and compliance in their software development.

3 Elaboration Chapter

3.1 Literature Review

With the rise of open source software in the recent decades, there is lots of scientific research available on different aspects of FOSS phenomena, such as open source development philosophy, motivation behind participating in FOSS projects, comparison of open source and proprietary software products and so on. Most of this research is not relevant for the purposes of this thesis. Scientific literature shedding light on usage of FOSS in commercial setting is scarcer. In his doctoral thesis (Hauge, 2010) Ø. Hauge identified six ways how organizations are adopting OSS:

- Deploying OSS products or tools in their operation environment as end users.
- Using OSS CASE tools in their software development.
- Integrating OSS components into their own or their clients' software systems.
- Participating in the development of OSS products.
- Providing OSS products.
- Using OSS development practices.

For this research "Integrating OSS components" is the focal topic with "Participating in the development of OSS products" and "Providing OSS products" also being somewhat relevant. The other three ways of organizational adoption of OSS are out of scope of this thesis.

A systematic literature review by M. Höst and A. Oručević-Alagić (Höst & Oručević-Alagić, 2011) investigates precisely the usage of open source software in commercial product development. Out of the four identified categories:

- Company participation in open source development communities
- Business models with open source in commercial organizations
- Open source as part of component based software engineering
- Using the open source process within a company

only two are in scope of this research: "Company participation in open source development communities" and "Open source as part of component based software engineering", with the latter being the research focus. Only articles concerning themselves with the focal research topic of this thesis were considered for literature review. Some of them touched the field of interest slightly while shifting the focus to different aspects. For example, in (Arhippainen, 2003) a case study conducted at Nokia on the use of third-party components is presented. Even though the report touches the integration of FOSS components, the focus lies on the usage of OTS in general.

In frames of this research work, it was a clear purpose to investigate in depth solely the works, focusing exactly on the area of interest, which is: recommendable practices for FOSS governance and license compliance in commercial setting. I.e. how companies can deal with the specifics of reusing open source code / components in the development of their (commercial) products most efficiently. In order to get the broader and more practice related view on the topic, whitepapers of prominent companies successful in the field of FOSS governance were taken into consideration next to scientific publications.

Twenty papers were selected for literature review. The full list of the papers is available in appendix. Based on the literature review, certain categories and sub-categories on the topic were gradually determined. The higher-level categories include “People”, “Processes”, “Tools”, “Policies” and “Strategy”. These categories are in line with previous research. R. Kemp (Kemp, 2010) mentions that FLOSS governance is a complex task and can be viewed as consisting of several building blocks, such as: the people context, the strategic context, the policy context and the process context. He also distinguishes an “achievements to date” context, which was not viewed as a separate category in frames of this research. Each of the categories contains multiple sub-categories, which will be discussed further. The determined categories composed a comprehensive literature review framework, which was filled out for each of the reviewed papers. The most important findings are summarized further in this chapter and the complete results table can be found in appendix.

The literature suggests that the first step on the way to successful FOSS governance is the definition of company’s FOSS policy (Haddad, 2010d), (Mutkoski, 2004), (Kemp, 2010), (BearingPoint, 2012). A policy is a set of rules for the use and management of FOSS in the organization (Haddad, 2010b). While there are examples of sample FOSS policies available (The Linux Foundation, 2012b), [4], they can only serve as supporting materials, since there is no “one-size-fit-all” FOSS policy. It depends strongly on the individual characteristics of each company and has to be aligned with company’s business model and FOSS strategy (Kemp, 2010), (Mutkoski, 2004). It is recommendable to start with reaching a high-level consensus in the form of FLOSS strategy, covering every business influencing aspect, while being aligned with other statements of organizational strategy (Haddad, 2010b), (Fendt et al., 2016), (Kemp, 2010). The experience report (Black Duck, 2015) points out that the efficient usage of OSS combined with mitigating the associated risks was only possible due to clearly defined OSS strategy. (Kemp, 2010) provides an example of how FLOSS Strategy statement may look like.

The purpose of FOSS policy is to define how the company is going to leverage the benefits of using open source software, while mitigating the associated risks, in a way that is consistent with the company’s business model and strategic goals. FOSS policy is also called to provide guidance for company’s employees and further FOSS governance process implementation (Koltun, 2011). It should define how the company should act in the common scenarios or issues regarding FOSS usage (Mutkoski, 2004). It is worth mentioning that according to (Black Duck & BearingPoint, 2013): “having a policy against open source is impractical and places you at a competitive disadvantage”. According to (Kemp, 2010), a good FOSS policy should: be clear and brief; be event driven; set out criteria and decision points for FLOSS use; define the information to be collected and tracked. Once FOSS policy is formulated, it needs to be enforced throughout the enterprise (Haddad, 2010d).

One of the important points in a company’s FOSS policy can be creating and maintaining good relationship with FOSS community, since the community provides source code, support, documentation and testing for the FOSS software (Haddad, 2010d). Such activities as reaching out and supporting FOSS organizations; participation in FOSS events and conferences are helpful in building relationship with FOSS organization. It is recommendable to set up a company website dedicated for distribution of FOSS packages and providing an external communication channel with the world and FOSS community. Email contact information for FOSS-related questions should be presented on this website (Haddad, 2010b).

Many companies have their employees as members of FOSS communities and this proves to be a good practice (Chang et al., 2010). Contributing changes back to the open source project can be beneficial, since it makes it easier to include updates / bug fixes of the open source component. However, this might cause leaking of essential organizational IP (Höst et al., 2011). In this regard, the company’s FOSS policy should clearly regulate the following aspects (Mutkoski, 2004):

- employees participation in external community projects
- contributing back to FOSS projects
- releasing company code under open source license
- setting up / sponsoring / leading FOSS projects

In order for the organizational FOSS policy to take action, it is crucial to bring awareness to the company's employees and ensure a good understanding. Therefore, training is an essential integral part of the effective FOSS governance (Haddad, 2010d). The majority of the reviewed papers emphasize the importance of trainings. (Haddad, 2011b) and (Koltun, 2011) point out that all relevant stakeholders, i.e. all staff that come into contact with FOSS either on procurement, development or shipping stage, need to be sufficiently educated on FOSS policies. In (Haddad, 2010b) the examples of formal and informal kinds of trainings are outlined. (Mutkoski, 2004) and (BearingPoint, 2012) highlight the importance of tailoring the trainings according to specific employee roles.

Another mechanism to support FOSS policy implementation is providing clear practical guidelines covering, for example, the description of most commonly used FOSS licenses (Haddad, 2010b), whitelists and blacklists of approved/denied FOSS packages and licenses (BearingPoint, 2012), common mistakes and how to avoid them (Haddad, 2010c). It is recommendable to host an internal FOSS web portal, where FOSS policies, guidelines, FOSS related publications and presentations are easily accessible for all employees. Other forms of internal communication beneficial for FOSS policy enforcement are all-hands meetings where executives can show their support to FOSS policy and issuing a periodical newsletter with news around FOSS. (Haddad, 2010b).

Defined FOSS policies should be implemented in form of processes, which have to be efficient and light weight (Haddad, 2010b). The FLOSS processes have to take the strain of FLOSS governance (Kemp, 2010). It is crucial that the overhead of the process is outweighed by the benefits that the process yields. One of the biggest and most vital challenges regarding the process is to convince all the relevant stakeholder of the importance and value it holds, and make it clear that the only goal of the process is to ensure efficient produce of license compliant high quality products (Fendt et al., 2016). The most rewarding way to establish FLOSS governance is to integrate the FLOSS management activities into existing software development process (Haddad, 2010b), (BearingPoint, 2012). The papers (Haddad, 2011b), (Kemp, 2010) and (BearingPoint, 2012) suggest that the initial establishment of effective FLOSS governance should be run as any other development project, with sufficient planning, proper resource allocation, scheduling and progress tracking. Known project management best practices are relevant in this case, and the whitepaper of the Linux Foundation (Haddad, 2011b) provides an in-depth discussion of how they should be applied to foster successful FLOSS governance.

The process descriptions found in (Haddad, 2010c), (Haddad, 2011a), (Koltun, 2011), (Fendt et al., 2016), (Black Duck, 2016a), (Black Duck & BearingPoint, 2013) were analyzed and aggregated into a sample FLOSS governance process consisting of the following phases:

- Search
- Select
- Approve
  - Scanning the source code
  - Identifying and resolving any discovered issues
  - Performing license and / or architectural reviews
  - Deciding the approval for the software component
- Catalog
  - Registering the approved software component in the software inventory system
- Use and Satisfy obligations
  - Compiling all the obligations and notices related to the use of FOSS
  - Updating end user documentation to reflect FOSS usage in product
- Validate
  - Ensuring only authorized code is used
  - Performing a pre-distribution verification of all previous steps
- Distribute
  - Distributing any applicable software components and making available applicable notices
  - Performing post-distribution verifications
- Monitor

Recommendations for each of the process phases provided by literature are outlined further.

Search. The suitability criteria for software components have to be determined in advance – in frames of FLOSS policy. While searching for open source software, it is advisable to start with well-known FOSS projects. It is important to take the maturity of the community into account as well as to leverage the insider knowledge of the employees who are active in the community, if any (Höst et al., 2011). In order to achieve efficient search and foster FOSS reuse, it is recommended to employ a searchable knowledge base of open source code, containing the associated metadata (BearingPoint, 2012). Insights into successful usage of such tools are provided by (Black Duck, 2016b) and (Black Duck, 2015).

Select. It is vital to assure that only suited components are selected for integration. Common selection criteria include: programming language, good code quality, security, maintainability, quality of documentation, rather large and robust community, fast reaction on bug reports. Except for technical criteria and those connected with the characteristics of the FOSS community, it is important to take legal aspects into account, i.e. the constraints posed by licenses (Fendt et al., 2016), (Höst et al., 2011). It is advisable to look for the components that can be considered “ad hoc standard”, i.e. they are used in many products of the given kind (Höst et al., 2011). According to (BearingPoint, 2012), a good solution is to maintain a repository of the approved components and oblige developers to evaluate the fit of the components in the repository before suggesting new ones. This approach will increase the software architecture’s quality and decrease heterogeneity, while saving the effort for future monitoring of selected components.

Approve. It is advised to mandate developers to fill in a request form for every FOSS they want to include in company’s software product and to ensure that only approved FOSS components are accepted into build system (Haddad, 2010d), (Haddad, 2010b). The request form should include description of the FOSS component, open source project behind it and the intended use of FOSS in question (Haddad, 2010c). The Linux Foundation provides a sample template for approval request form (The Linux Foundation, 2012a). An incoming request from developers for FOSS usage should initiate a source code scan with automated source code analysis tool. It is also important to periodically conduct scheduled full platform scan (Haddad, 2011a). Regular source code scans provide a possibility to identify used FLOSS early in the development process, even in case programmers failed to submit a request for its usage, and by doing this to avoid possible extra effort caused by untimely FLOSS identification in the product code base (Protecode, 2012), (Link, 2010).

Once the core FLOSS governance team received a request for FLOSS usage and performed the source code scanning, they have to review the request in accordance with company's FLOSS policy, taking into account licensing aspects and eventually supporting the decision making process with architecture / link analysis reviews (Haddad, 2010a). It is crucial to establish an efficient mechanism for answering developer requests to make sure the requests are taken care of early in the development process. It is reasonable to centralize the task of reviewing such requests, at least for the initial phase of FLOSS governance establishment (Mutkoski, 2004). According to (Black Duck, 2016a), the approval process may easily become a bottleneck. The authors point out that with today's agile development methodologies, it is crucial to automate the approval process in order to achieve transparency and efficiency.

Catalog. After the approval, the approved FOSS usage in corporate products should be registered in software inventory system and uniquely tracked for future references (Haddad, 2011a), (Fendt et al., 2016). The goal of this activity is to constitute a database of already approved FOSS with related valuable information and facilitate internal knowledge sharing on the component's usage (Fendt et al., 2016). Cataloging approved components motivates the developers to consider reuse of already approved components prior to searching for the new ones, which considerably speeds up the development process. Moreover, it helps to increase standardization while reducing heterogeneity, product complexity and the maintenance effort for FOSS deployment. Similarly to approved FOSS components stored in a repository, a list of approved, conditionally approved and rejected FOSS licenses should be managed (BearingPoint, 2012).

Use and Satisfy Obligations. The approval and cataloging are followed by the use of the FOSS component in product development. On this stage, it is important to take care of satisfying license obligations. Normally, these include providing "OSS declaration" document, containing copyright and authorship acknowledgements along with copies of used components' licenses (Fendt et al., 2016). In some cases it is necessary to make the source code of FOSS available to the end user (Haddad, 2010a). In case modifications were made to the open source code, it is necessary to decide whether to contribute the changes back to community (Höst et al., 2011). It is vital to make sure that all licenses obligations were satisfied prior to making a product public, since it is a legal responsibility of the distributor – to ensure all license requirements are met (BearingPoint, 2012).

Validate. Prior to product distribution, it is necessary to verify the accomplishment of all the process steps, to assure that:

- • The code only contains approved FOSS components.
- • All necessary compliance actions are performed to satisfy license obligations.

Similarly to approval stage of the process, verification also includes using of code scanning tools to audit the full product code (Protecode, 2012), (Haddad, 2010a). The main target of this process step is to ensure that the approved FOSS code is exactly the same code that is used to build the product (BearingPoint, 2012).

Distribute and Monitor. After FOSS is acquired, approved, registered, validated and deployed, and the post-distribution verifications are over, it is critical to not lose sight of the adopted open source components and continue to monitor them over the full lifecycle. Even after deployment, issues can arise, such as bug fixes or security vulnerabilities. In this case it is important to keep up the visibility of which components are deployed where, to be able to solve potential problems efficiently (BearingPoint, 2012).

Establishing FLOSS governance, it is necessary to remember that there are two entry points for FOSS into a product: developer and supplier (BearingPoint, 2012). While the practices outlined above provide guidance on how to manage the "developer" entry point, it is crucial to realize that similar practices must be applied to third party supplied software, since the distributor is responsible for all the code contained in the product. It is important to adjust supply chain procedures to assure FOSS license compliance of acquired components (Haddad, 2010c). A best practice is to request detail FOSS usage declarations from the binary components suppliers with information about integrated open source components and the measures taken to fulfill the license requirements (Fendt et al., 2016). The paper (Koltun, 2011) by the Linux Foundation focuses solely on description of compliance best practices with regard to supplied software.

In order to be effectively implemented the process dimension must be supported by "People" and "Tools". According to (Kemp, 2010), in order to tame a company's open source usage and ensure effective FLOSS governance, a number of stakeholder groups should be involved, while integrating and cooperating in a supportive and positive way. The majority of reviewed papers suggest the need to establish a core and extended FLOSS governance team, as well as to assign a FLOSS Compliance Manager (also known as Compliance Officer, Director of Open Source, etc.) - the head of the core FLOSS governance team.

Core team, called Open Source Review Board (OSRB) in (Haddad, 2010c), should consist of Legal Representative(s), Engineering and Product Team Representative(s) and Compliance Officer. The OSRB reviews and approves FLOSS usage requests and serves as steering committee for company's FLOSS strategy (Haddad, 2010d). From strategic point of view it is crucial to gain executive C-level commitment to FLOSS governance and compliance (Haddad, 2010b), (Kemp, 2010), (Black Duck, 2014). The extended team should be built of various individuals from different departments, who contribute to FLOSS governance on as-needed basis, according to the tasks assignments from the core team (Haddad, 2010c). According to (Fendt et al., 2016), a beneficial approach for big companies would be for the core team to concentrate on high level issues, such as FLOSS strategy and policies definition, setup of the processes and infrastructure, organization of trainings, while a separate so called Clearance Team takes care of the operational aspects on a business segment level.

Roles and responsibilities of OSRB, extended team and Compliance Officer as well as the required competences for these roles are described in great detail in the whitepaper of the Linux Foundation (Haddad, 2010c) and further elaborated in the chapter 3 of the book published by the Linux Foundation (Haddad, 2016). Several papers, such as (Fendt et al., 2016), (Haddad, 2011b), (Haddad, 2010b), emphasize the importance of the proper staffing and capacity of the FLOSS governance teams, since they can easily become a bottleneck in the process. Experience reports from Samsung (Chang et al., 2010) and Siemens (Fendt et al., 2016) provide an insight into how these companies successfully shaped their FLOSS governance teams.

According to (BearingPoint, 2012), it is impossible to take a firm grasp on FOSS governance with manual processes. The complexity of the associated tasks is too sheer, making manual methods doomed for incompleteness, inconsistency and non-sufficient speed to keep up with the modern agile development lifecycle. Therefore, there is a strong need for automation (Haddad, 2011b), (BearingPoint, 2012), (Black Duck, 2016a). The absolute majority of the reviewed papers emphasize the importance of using source code-scanning tools. These tools identify open source code included in the code base and their licenses. This is realized through matching of the source code with a knowledgebase of existing open source code. Code-scanning tools are normally used on the approval and verification stages of the FLOSS governance process (Fendt et al., 2016), (BearingPoint, 2012), (Chang et al., 2010).

Some papers suggest using project management tools to assist in FLOSS governance for recordkeeping, issue tracking, scheduling and reporting on FLOSS governance tasks (Haddad, 2011b), (Haddad, 2010c). In order to maintain a clear overview of the FLOSS adoption and foster effective reuse, it is recommendable to employ software inventory tool or OSS component management system (Fendt et al., 2016), (Chang et al., 2010). Other advised tools mentioned in the literature include Bill of materials (BoM) difference tool; dependency tracking tool; binary scanning tool; code janitor tool; source code peer review tool; license to requirements mapper; product FOSS documentation generator; product FOSS code collector (Haddad, 2010c), (Fendt et al., 2016).

The most recent papers mention holistic solutions, uniting all of the tools mentioned above and enabling companies to streamline the entire FOSS process. One example of such tool is Black Duck software suit, which automates search, selection, approval, validation and monitoring of FLOSS in the product development. It enables companies to integrate FLOSS governance tasks into software development process in an effective way, while avoiding unnecessary delays and developers‘ resistance to the process overhead (Black Duck, 2016a). Whitepapers (Black Duck, 2015) and (Black Duck, 2014) provide the success stories of the Black Duck customers who have managed to take a grasp on their FLOSS governance with the help of the process automation.

3.2 List of best practices

3.2.1 Policies

Respect the license

Name	Respect the license
Actor	Every employee
Context	You are using / planning to use open source components in your software development
Problem	How to ensure compliance of your software?
Solution	Be aware of the different open source licenses and ensure compliant usage of open source in software development. Use *Create compliance guidelines* and *Conduct trainings* practices to provide preventive measures. Include licensing aspect into selection criteria in frames of *Select wisely* practice. Apply *Approve* and *Using code scanning tools* practices to ensure that only approved open source software is used in the products and thus assure software compliance. Although, you should not rely solely on scanning tool, but groom the understanding and respect to open source licenses in your employees instead. In case legal department concludes that a certain license cannot be used in a certain software piece – an alternative solution must be selected. You should try to detect any incompliances as early in the development process as possible. However, even if detected in the late stage – the incompliances must be resolved before software distribution. Conduct regular software audits to foster license compliance. You should keep track of all the used open source components, their licenses and corresponding obligations. After software distribution, the license obligations must be fulfilled.
References	Interviews: 3, 5; Literature: every reviewed paper

Conduct trainings

Name	Conduct trainings
Actor	FLOSS governance team / Open source director
Context	You are using open source code / components in your software product development
Problem	You want to raise awareness among your employees and foster FLOSS compliance
Solution	Conduct trainings on open source matters for your employees in order to educate them on the topic and achieve a mindset for compliance issues. The options for training format include sending employees to formal external trainings, incorporating FLOSS training into standard trainings, using web-based training solutions. You can integrate FLOSS governance and compliance training as a separate agenda item in your regular team trainings, e.g. project onboarding or project management training. At first you may rely on training materials of more experienced external organizations and eventually as your maturity level improves, develop your own training materials. Materials of Linux Foundation on open source governance and compliance are a good starting point for training development.
References	Interviews: 1, 2, 3, 4, 5; Literature: (Haddad, 2010d), (Haddad, 2010c), (Haddad, 2010b), (Haddad, 2011b), (Koltun, 2011), (Fendt et al., 2016), (Mutkoski, 2004), (Kemp, 2010), (BearingPoint, 2012), (Chang et al., 2010)

Create compliance guidelines

Name	Create compliance guidelines
Actor	Legal department / FLOSS governance team / Open source director
Context	You are using open source code / components in your software product development
Problem	How to speed up the FLOSS governance and approval process and foster correct open source use among your employees?
Solution	Develop compliance guidelines explaining the common rules of open source usage in the company, e.g. giving an introduction into the concept of open source and different license types, providing the list of licenses that are allowed for usage. Usually compliance guidelines are composed by company's legal department. The guidelines document should not stay static, but evolve over time as new licenses or open source components come up. The guidelines do not have to be restricted only by legal licensing aspect. They can also include description of the company's FLOSS governance process and internal best practices. A recommendable format for such guidelines is a Wiki-like system. In this case, maintaining this document, which constantly evolves over time, is the responsibility of Open source director.
References	Interviews: 1, 4, 5; Literature: (Haddad, 2010c), (Haddad, 2010b), (Koltun, 2011), (BearingPoint, 2012), (Chang et al., 2010)

Involve in FOSS communities

Name	Involve in FOSS communities
Actor	Software engineer
Context	You are using open source code / components in your software product development
Problem	You want to build a good relationship with open source communities
Solution	Have your employees involved in FOSS communities. This involvement provides a source of valuable information about the communities, which can be used for *Select wisely*. Having employees as members of open source community, a company can rely on them for the communication with the community and eventually even influence the course of action of the developments in the project. Employing this practice directly supports *Contribute back* practice. You should get aligned with Linux Foundation – one of the most important organizations in the modern open source world. You may become a member of Linux foundation or a comparable organization, which provides a platform for discussing FLOSS related matters with other involved companies and share experiences.
References	Interviews: 1, 2, 3, 4, 5; Literature: (Haddad, 2010d), (Haddad, 2010b), (Fendt et al., 2016), (Mutkoski, 2004), (Höst et al., 2011), (Chang et al., 2010), (Black Duck & BearingPoint, 2013), (Black Duck, 2015)

Contribute back

Name	Contribute back
Actor	Software engineer
Context	You implemented some changes to the used open source component.
Problem	How do you ensure that you can leverage all the advantages of using open source components over self-developed components?
Solution	In order to be able to leverage all the benefits of using open source components in software development, such as software updates and bug fixes provided by the community, it is recommendable always to contribute your changes back to community. You should not fork open source projects. You should contribute your changes back since it provides benefits in form of code maintenance and further improvement by community. You should pay attention to the viability of contributing back to the community during selection process. You should always keep your core competence in mind, when contributing to open source projects and not let any key intellectual property out. In order to ensure the technical quality and legal compliance of the code as well as to avoid problem with IP and patents, you can use code review mechanism before submitting the contribution to community. A recommendable practice is to develop a checklist of important points in cooperation with legal department, which the reviewer can use.
References	Interviews: 2, 3, 4, 5; Literature: (Link, 2010), (Mutkoski, 2004), (Höst et al., 2011), (Chang et al., 2010), (Black Duck & BearingPoint, 2013), (Black Duck, 2015)

Ensure 3rd party software compliance

Name	Ensure 3rd party software compliance
Actor	Engineering manager
Context	You are using third-party components in your software product development
Problem	How to ensure compliance of supplied third-party code?
Solution	Remember that there are two entry points for FOSS into a product: developer and supplier. It is necessary to employ mechanisms to ensure that the supplied software is compliant in terms of open source usage. In order to mitigate the risks of supplied software non-compliance you should strictly regulate the responsibility for the supplied code compliance through signing corresponding contracts with their suppliers. You should also require an SPDX document with each software delivery.
References	Interviews: 4, 5; Literature: (Haddad, 2010c), (Haddad, 2010b), (Haddad, 2011a), (Haddad, 2010a), (Koltun, 2011),(Fendt et al., 2016), (Link, 2010), (Mutkoski, 2004), (Kemp, 2010), (BearingPoint, 2012), (Chang et al., 2010), (Black Duck, 2014)

3.2.2 Processes

Select wisely

Name	Select wisely
Actor	Engineering manager / software engineer
Context	You have decided to use open source in your software product development to solve particular problem.
Problem	How do you select an open source component to use?
Solution	The first criteria for selection of OSS is that it solves the given problem in a technically appropriate way. Another important step in the selection process is checking the license's compatibility with your final product. In case there are several viable options available, it is recommendable to evaluate the corresponding FOSS communities. It is advisable to choose large diverse communities and avoid communities, which are driven by a single company, as well as inactive communities. Robust and active communities are more likely to provide support for the component in the future. It is much easier to assess the open source communities when you *get involved in FOSS communities*. You should employ an evaluation framework applied to all the viable options during selection process. It should take into account technical characteristic, licensing as well as cost-risk analysis of using each component, e.g. how justified is each component's use from the economic point of view.
References	Interviews: 1, 2, 3, 4, 5; Literature: (Fendt et al., 2016), (Höst et al., 2011), (BearingPoint, 2012), (Chang et al., 2010), (Black Duck, 2016a), (Black Duck & BearingPoint, 2013), (Black Duck, 2016b), (Black Duck, 2015), (Black Duck, 2014)

Approve

Name	Approve
Actor	FLOSS governance team / Engineering manager
Context	An open source component was selected for use in your product.
Problem	How do you ensure that this component is suitable and compliant?
Solution	Integrate an approval mechanism in the FLOSS governance process. After you *Select wisely*, the selected component should be reviewed and approved before it can be used. Often this is a responsibility of the centralized FLOSS governance team. You want to make sure that approval does not become a bottleneck in the process. This can be achieved through proper team staffing, delegating some part of the duties to the extended team, educating the engineers, providing clear compliance guidelines. You can *Use code scanning tools* to speed up the approval process. In case a request cannot be approved – it is crucial to communicate the reasons clearly to avoid conflicts and raise the open source awareness among software engineers.
References	Interviews: 1, 3, 5; Literature: (Haddad, 2010d), (Haddad, 2010c), (Haddad, 2010b), (Haddad, 2011b), (Haddad, 2011a), (Haddad, 2010a), (Koltun, 2011), (Fendt et al., 2016), (Mutkoski, 2004), (Kemp, 2010), (BearingPoint, 2012), (Black Duck, 2016a), (Black Duck & BearingPoint, 2013), (Black Duck, 2016b), (Black Duck, 2014), (Protecode, 2012)

Catalog

Name	Catalog
Actor	Software engineer / Engineering manager
Context	An OSS component was selected, approved and used in the product development.
Problem	How do you keep track of all the used components and foster efficient reuse?
Solution	Establish a repository where all the used OSS components are stored along with relevant descriptions, e.g. component characteristics, license information, use context. After you *Select wisely* and *Approve* you should catalog the used element, i.e. add the relevant information about it into the repository specifically designed for this purpose. Best practice is to use a template for the OSS component description. Cataloging the components and their licenses can save a lot of time in selecting and approving the components.
References	Interviews: 1, 2, 5; Literature: (Haddad, 2011a), (Haddad, 2010a), (Fendt et al., 2016), (Mutkoski, 2004), (Kemp, 2010), (BearingPoint, 2012), (Chang et al., 2010), (Black Duck, 2016a), (Black Duck & BearingPoint, 2013), (Black Duck, 2016b), (Black Duck, 2015), (Protecode, 2012)

3.2.3 People and tools

Have a centralized FLOSS governance team

Name	Have a centralized FLOSS governance team
Actor	Top management
Context	You are using open source code / components in your product development. You have a big number of developers working on the code and not all of them have a lot of experience with FLOSS.
Problem	You need to ensure that FLOSS is used rightfully and effectively.
Solution	Have a centralized FLOSS governance team. Typically, the centralized team consists of members of legal department, engineering and business representatives (e.g. product managers). The responsibilities of the centralized team include ensuring license compliance, approving open source components for use, checking and making decisions on contributing back to community. This team does not necessarily have to be a formal board. It may be enough to have people working on the centralized FLOSS governance team for a limited part of their working time (e.g. 20%).
References	Interviews: 1, 2, 3, 5; Literature: (Haddad, 2010d), (Haddad, 2010c), (Haddad, 2010b), (Haddad, 2011a), (Koltun, 2011), (Fendt et al., 2016), (Mutkoski, 2004), (Kemp, 2010), (BearingPoint, 2012), (Chang et al., 2010), (Black Duck, 2014)

Assign an open source director

Name	Assign an open source director
Actor	Top management / FLOSS governance team
Context	You are using open source code / components in your product development. You have a big number of developers working on the code and not all of them have a lot of experience with FLOSS.
Problem	You need a person responsible for effective, rightful and strategically beneficial use of FLOSS.
Solution	Assign a person responsible for FLOSS use in the company. This person is typically the head of the centralized FLOSS governance team. This position is often called “Open source director”, also known as “Compliance manager” or “Compliance officer”. Open source director’s responsibilities include: tracking what is happening in the open source world, serving as an interface for all the business functions (engineering, legal, marketing) when it comes to the open source questions, ensuring open source license compliance and strategic use, defining open source policy and guidelines.
References	Interviews: 3,5; Literature: (Haddad, 2010c), (Haddad, 2010b), (Haddad, 2011b), (Koltun, 2011), (Kemp, 2010), (Chang et al., 2010), (Black Duck, 2014)

Establish an extended FLOSS governance team

Name	Establish an extended FLOSS governance team
Actor	FLOSS governance team
Context	You are using open source code / components in your product development. You already Have a centralized FLOSS governance team.
Problem	You need to solve all the issues with FLOSS governance efficiently and do not want the centralized FLOSS governance team to become a bottleneck.
Solution	Have an extended team next to the centralized FLOSS governance team. The members of the extended team are employees of different company’s departments and are working on the specific FLOSS governance issues on an on-demand basis. The examples of such issues are: a previously unknown license, which needs particular attention of legal department; a choice of an open source component to use which will affect the product’s features, thus needs to be reviewed by business stakeholders. You should involve more employees to FLOSS governance matters on an on-demand basis, in order to avoid turning the centralized FLOSS governance team into a process bottleneck. Building up the expertise among engineers may take a lot of burden away from the centralized team.
References	Interviews: 1, 3, 4, 5; Literature: (Haddad, 2010c), (Haddad, 2010b), (Fendt et al., 2016), (Kemp, 2010), (Chang et al., 2010)

Use code scanning tools

Name	Use code scanning tools
Actor	Engineering manager
Context	You are using open source code / components in your product development.
Problem	You need an effective way to keep track of open source components and ensure license compliance.
Solution	Use code scanners in order to keep track of the used open source components and assure license compliance of the software. Using code scanners should not be a substitute for responsible use of open source by software engineers, but rather serve as an aid in verifying the compliance of the software products. This approach is more efficient and precise than manual checks. Nevertheless, it may still require certain manual effort to go over the scanning results. Especially important is code scanning for 3rd party software compliance. Broadly used commercial code scanning tools are offered by Black Duck and Palamida. There are also open source tools available, e.g. Fossology, sw360.
References	Interviews: 1, 2, 3, 4, 5; Literature: (Haddad, 2010d), (Haddad, 2010c), (Haddad, 2010b), (Haddad, 2011b), (Haddad, 2011a), (Koltun, 2011), (Fendt et al., 2016), (Mutkoski, 2004), (Kemp, 2010), (BearingPoint, 2012), (Chang et al., 2010), (Black Duck, 2016b), (Black Duck, 2014), (Black Duck, 2016a)

3.3 Conclusion

The goal of this study was to develop a theory of industry best practices of FLOSS governance and compliance that can be used by software development companies in order to leverage the benefits of using open source and mitigate the associated risks. Case study research methodology was applied, semi-structured interviews were conducted with industry partners from five different companies. Suggested best practices were derived from raw data with the help of qualitative data analysis methods and tools. The results of this research include a handbook of best industry practices of FLOSS governance and compliance formulated in the form of best practice patterns as well as a summary of the key findings from case studies. The identified best practices can be employed by software development companies striving for successful FLOSS governance and compliance.

3.4 Acknowledgements

I would like to acknowledge the Open Source Research Group at FAU, in particular Prof. Dr. Dirk Riehle, Nikolay Harutyunyan and Andreas Kaufmann who supported me throughout the entire research process by helping in industry partners selection, interview organization and conduction as well as providing me access and support to a qualitative analysis tool – QDAcity – and feedback on my continuous work.

I would like to acknowledge our industry partners for their engagement, cooperation and highly valuable insights on FLOSS governance and compliance in an organization, which they kindly agreed to share during the interviews.

Appendix A Case Study Protocol

1. Background

a) identify previous research on the topic – literature review following the methodology described in (Webster & Watson, 2002) on recommendable practices for FOSS governance and license compliance in commercial setting was conducted and contributed to research question and interview questions formulation.

b) define the main research question being addressed by this study –“ What are the best Industry Practices of FLOSS Governance and Compliance?”

2. Design

a) identify whether single-case or multiple-case and embedded or holistic designs will be used, and show the logical links between these and the research questions – Multiple-case and embedded design is used, because there are five cases under investigation and each case touches multiple best practices.

b) describe the object of study – FLOSS governance in five different companies.

c) identify any propositions or sub-questions derived from each research question and the measures to be used to investigate the propositions – four categories of best practices were derived during literature review: People, Policies, Processes, Tools. Their investigation is conducted with means of qualitative data analysis. Each category includes a separate subset of codes.

3. Case Selection

a) Criteria for case selection – diversity based on the following characteristics: type of product / service; type of customer; market position; size; maturity, with size and maturity being the central ones.

4. Case Study Procedures and Roles

a) Procedures governing field procedures – in total there were siz interviews scheduled with industrial partners. Some of them were conducted during personal visits, others – online. Some of the interviews were conducted by Nikolay Harutyunyan. The list of interview questions presented in Appendix B was called to direct the interviews, while adapting to the interviewee’s behavior on the fly.

5. Data Collection

a) identify the data to be collected –The main source of data collected from the case studies were semi-structure interviews with partners of different open source governance related roles in the given company. Interview questions are presented in Appendix B.

b) define a data collection plan – each interview was recorded and transcribed. Other relevant data about each company was additionally pulled in. Data collection for the next cases succeeded in parallel to analysis of already collected data.

c) define how the data will be stored – The interview recordings were stored digitally as audio files; the interview transcriptions – as text files.

6. Analysis

a) identify the criteria for interpreting case study findings – In preparation phase of the case study, four categories of best practices were derived: people, policies, processes and tools. Each category had a subset of related codes, which were iteratively developed during qualitative data analysis procedures. The results of the research were aggregated based on the coding.

b) the analysis should take place as the case study task progresses – after collecting data for one of the case studies – analysis started immediately, while continuing data collection for the next case in parallel.

7. Plan Validity

a) general: check plan against Höst and Runeson's (2007) checklist items for the design and the data collection plan – The checklist was used to assure the quality and validity of the conducted analysis. Filled checklist can be found in Appendix C.

b) construct validity -show that the correct operational measures are planned for the concepts being studied. Tactics for ensuring this include using multiple sources of evidence, establishing chains of evidence, expert reviews of draft protocols and reports – The systematic use of qualitative data analysis techniques ensures the connection of the research results to the raw data aiding chains of evidence establishment.

c) internal validity -show a causal relationship between outcomes and intervention/treatment (for explanatory or causal studies only). – The code system defined in frames of qualitative data analysis highlighted the interconnections between different sources of information and best practices, while ensuring internal validity.

d) external validity –identify the domain to which study finding can be generalized. Tactics include using theory for single-case studies and using multiple-case studies to investigate outcomes in different contexts. – Multiple case study design assured external validity. With five case studies conducted at different specifically selected companies, it was possible to formulate a theory of best practices that can be applied by any company in the domain of FLOSS governance.

8. Study Limitations

Specify residual validity issues including potential conflicts of interest (i.e. that are inherent in the problem, rather than arising from the plan). – The biggest limitation that arose during the case studies was that most of the documents related to the investigated topic were too confidential to get direct access to them, so the main source of information were interviews with the companies' employees.

9. Reporting

Identify target audience, relationship to larger studies (Yin, 2013) – The target audience of this thesis are companies using FLOSS in their software development and striving for effective FLOSS governance. The research contributes to the larger research domain of using FLOSS in commercial setting.

10. Schedule

Give time estimates for all of the major steps: Planning, Data Collection, Data Analysis, Reporting. Note Data Collection and Data Analysis are not expected to be sequential stages – Out of the six months devoted to this master thesis two months were spent on literature review and case study preparation, three months on data collection and analysis and one month on reporting.

11. Appendices

a) Validation: report results of checking plan against Höst and Runeson's (2007) checklist items – Can be found in Appendix C.

b) Divergences: update while conducting the study by noting any divergences from the above steps. –There were no major divergences from the above outlined steps.

Appendix B Interview questions

Interviewee Context

1. What is your position in the company?
2. What do your responsibilities include and how are you involved with FLOSS at your company?

FLOSS in the organization

3. How does your company use FLOSS?
4. Does your company use FLOSS in the product development?
5. Why are you using FLOSS in your products? Which benefits does it bring you?
6. How long have you been using open source components in your products?
7. Do you have established mechanisms for managing the use of FLOSS?
    7.1. what specific mechanisms?
    7.2. processes?
    7.3. tools?

Strategy & Policy

8. Do you have a documented FLOSS strategy/program?
    8.1. What are its most important points?
    8.2. Could we get access to this document?
9. Do you have a documented FLOSS policy?
    9.1. What are its most important points?
    9.2. Could we get access to this document?
10. Is creating and maintaining good relationship with FOSS community a part of your FLOSS policy? If so, how do you achieve this?
11. Do you have the company employees as members of FOSS communities? Is this activity encouraged / regulated by the company? How?
12. Do you contribute the modified code back to FOSS communities? Why / why not?
13. Have you ever led / set up / sponsored FOSS projects?
14. Do you provide FOSS trainings for your employees?
    14.1. What kind of trainings (formal, informal, what is the content etc.)?
    14.2. Do you organize trainings yourself or take help from consultants, e.g. Linux Foundation?
    14.3. How often?
    14.4. For which employee groups?
    14.5. Success rates?
15. Do you have guide-lining documents for FOSS usage?
    15.1. Do you have a mechanism to make these documents easily accessible?
    15.2. Do you host an internal FOSS web portal?
    15.3. Do you have an internal FOSS newsletter?

People

16. Who is responsible for making a final decision on which open source components to use?
17. Do you have a centralized FLOSS governance team?
    17.1. Who are the members of this team? What competences are important?
    17.2. What are the responsibilities of the core team? Is being a member of the team a full-time occupation?

    17.3. Is the centralized team responsible for final decision making / approval of which open source code / components to use?

18. Do you have a position of Compliance Officer (Director of Open Source) or similar (the head of the core team)?
    18.1. Which competences are important for this role?
    18.2. What are the responsibilities of this person?
19. Do you have an extended team / separate teams pro business unit?
    19.1. Who are the members of this team?

Processes & Tools

20. Do you have a documented FLOSS usage process?
    20.1. Could we get access to this document?
    20.2. What are the process phases?
21. Who and how searches for open source components to use in a product?
    21.1. Is the search process regulated in any way?
    21.2. Are any tools used to foster the search process, e.g. a repository of FOSS approved for use?
22. Who and how selects open source components to use in a product?
    22.1. Is the selection process regulated in any way?
    22.2. Are there any defined selection criteria?
    22.3. Are any tools used to foster the selection process, e.g. a repository of FOSS approved for use?
23. Do you have an approval mechanism for FOSS usage in place?
    23.1. How does it work?
    23.2. Who is responsible for reviewing approval requests and granting approvals?
    23.3. Are any tools used to streamline this process?
24. Do you use code scanning tools?
    24.1. What are the triggers for using these tools?
25. How do you keep track of approved / used FOSS components?
26. Do you have a process for a license compliance check?
    26.1. list of licenses allowed for use?
    26.2. using tools to check license compliance?
27. How do you satisfy license obligations?
    27.1. updating user documentation?
    27.2. distributing open source code?
    27.3. automatization?
28. How do you ensure compliance of your products in terms of FOSS usage before distribution?
    28.1. validation before distribution?
29. How do you ensure that 3rd party software you acquire from suppliers is compliant?
30. How do you monitor used FOSS components after deployment?
31. Do you employ a holistic solution to connect all the tools used to support the process on each phase and streamline (automate) the entire process?

Other questions

32. Are there other FLOSS governance and compliance best practices you would like to mention?
33. Is there anything else you would like to mention?

Optional questions

34. What is your academic / professional background?

Note: The “how” questions imply that the interviewee will bring up the tools used for the mentioned activities, if any.

Note: All the questions concern themselves with the common practices of conducting FLOSS governance. The question “how to get there?” (i.e. the practices for initial establishment of FLOSS governance) is intentionally left out of scope for this work, in order to narrow down the research focus.

Appendix C Researcher’s Checklist

Case Study Design

1. What is the object of study? – FLOSS governance and compliance practices at five different companies.
2. Is a clear purpose/objective/research question /hypothesis/proposition defined upfront? – This exploratory case study is devoted to answering the following research question: ”What are the best industry practices of FLOSS governance and compliance?”. More detailed information is presented under “Research question” section.
3. Is the theoretical basis -relation to existing literature and other cases -defined? –Yes. Related Work section of the Research Chapter presents an overview of available literature on the topic and compares the literature review findings with the research implication of the conducted study.
4. Are the authors’ intentions with the research made clear? –Yes. The research intentions are presented in the Research Question section of the paper.
5. Is the case adequately defined (size, domain, process…)? –The detailed description of the case contexts is given in Used Data Sources section.
6. Is a cause-effect relation under study? If yes, is the cause distinguished from other factors? – each best practice pattern presented under List of best practices in the Elaboration chapter displays a cause-effect relation derived from collected data.
7. Will data be collected from multiple sources? Using multiple methods? – most of the data was collected through semi-structured interviews with industry partners from five different companies.
8. Is there a rationale behind the selection of roles, artefacts, viewpoints, etc.? – five companies for case studies were chosen through theoretical sampling based on certain criteria Detailed information is presented under Research Approach and Used Data Sources section of Research Chapter.
9. Is the integrity of individuals/organizations taken into account? – Each company represents an integral organization with its own practices.

Preparation for Data Collection

10. Is a protocol for data collection and analysis derived (what, why, how)? –The case study protocol can be found in Appendix A. It addresses all the major steps of the case study research.
11. Are the planned methods and measurements sufficient to fulfil the objective of the study? – Computer aided qualitative data analysis was chosen as a method of analysis of collected data. This method is sufficient for achieving the objective of the study – deriving best practices of FLOSS government and compliance from the collected data.
12. Is the study design approved by a review board, and has informed consent obtained from individuals and organizations? –The study design was approved by the responsible university chair and the informed consent is obtained from both individuals and organizations participating in case studies.

Collecting Evidence

13. Are data collected according to the protocol? –Yes.
14. Are data recorded to enable further analysis? –Yes.
15. Are sensitive results identified (for individuals, organization or project)? –Yes.

16. Are the data collection procedures well traceable? –Yes, the list of interview questions is presented in Appendix B and all the collected data is stored both as audio recordings of the interviews and text files of interview transcriptions.
17. Do the collected data provide ability to address the research question? –Yes.

Analysis of Collected Data

18. Is the analysis methodology defined, including roles and review procedures? –Yes, the qualitative data analysis methodology is used, with the help of the tool QDAcity.
19. Is a chain of evidence shown with traceable inferences from data to research questions and existing theory? –Yes, each best practice presented in Elaboration chapter includes references to interviews and literature sources. Some best practices and the Results section of Research Chapter contain interview quotes.
20. Are alternative perspectives and explanations used in the analysis? –Yes, each best practice is coded and includes references to the alternative perspectives.
21. Are there clear conclusions from the analysis, including recommendations for practice/further research? –Yes, the list of best practices can be used by companies striving for successful FLOSS governance. Section 2.6 presents a summary of the research results, while section 2.7 presents limitations, conclusions and outlook for future research.
22. Are threats to validity addressed in a systematic way? –The threats to validity are minimized using robust qualitative data analysis methods, case study protocol and case study checklist.

Reporting

23. Are the case and its context adequately reported? –Yes.
24. Are the research questions and corresponding answers reported? –Yes.
25. Are the data collection procedures presented, with relevant motivation? –Yes.
26. Are sufficient raw data presented? –Yes.
27. Are the analysis procedures clearly reported. –Yes.
28. Does the report contain conclusions, implications for practice and future research? –Yes.
38. Is the report suitable for its audience, easy to read and well structured? –Yes.

References

Arhippainen, L. (2003). Use and integration of third-party components in software development. *VTT Publications*, (489), 3–68.

BearingPoint. (2012). *FOSS Management Study*.

Bereton, P., Kitchenham, B., Budgen, D., & Li, Z. (2008). Using a Protocol Template for Case Study Planning. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*. University of Bari, Italy.

Black Duck. (2014). *Sita Builds Path to Success with Open Source Software Strategy*.

Black Duck. (2015). *Samsung Wins Big with Open Source Lifecycle Strategy*.

Black Duck. (2016a). *Taking Control of Open Source Software in Your Organization*.

Black Duck. (2016b). *The Enterprise IT Guide to Open Source Software Management*.

Black Duck, & BearingPoint. (2013). *Open Source Governance in Highly Regulated Companies*.

Chang, S., Lee, J., & Yi, W. (2010). A Practical Management Framework for Commercial Software Development with Open Sources. *E-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on*, 164–171. https://doi.org/10.1109/ICEBE.2010.82

Fendt, O., Jaeger, M., & Serrano, R. J. (2016). Industrial Experience with Open Source Software Process Management. *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, 180–185. https://doi.org/10.1109/COMPSAC.2016.138

Firestine, B. (2017). Celebrating nine years of GitHub with an anniversary sale. Retrieved from https://github.com/blog/2345-celebrating-nine-years-of-github-with-an-anniversary-sale

Free Software Philosophy. (n.d.). Retrieved from http://www.gnu.org/philosophy/open-source-misses-the-point.en.html

Haddad, I. (2010a). *Achieving FOSS Compliance in the Enterprise*.

Haddad, I. (2010b). *Establishing Free and Open Source Software Compliance Programs : Challenges and Solutions*.

Haddad, I. (2010c). *Free and Open Source Software Compliance : Who Does What*.

Haddad, I. (2010d). *Free and Open Source Software Compliance The Basics You Must Know*.

Haddad, I. (2011a). *A Five-Step Compliance Process for FOSS Identification and Review*.

Haddad, I. (2011b). *Keys to Managing a FOSS Compliance Program*.

Haddad, I. (2016). *Open Source Compliance in the Enterprise*. The Linux Foundation.

Hammond, J., Santinelli, P., Billings, J. J., & Ledingham, B. (2016). The Tenth Annual Future of Open Source Survey. Retrieved from https://www.blackducksoftware.com/2016-future-of-open-source

Harutyunyan, N. (2016). *Theory of User Experience Best Prac- tices in Software Product Lines*.

Hauge, Ø. (2010). *Adoption of Open Source Software in Software-Intensive Industry* (Vol. 35). https://doi.org/10.1007/978-3-642-14616-9_35

Höst, M., & Oručević-Alagić, A. (2011). A systematic review of research on open source software in commercial software product development. *Information and Software Technology*, *53*(6), 616–624. https://doi.org/10.1016/j.infsof.2010.12.009

Höst, M., Oručević-Alagić, A., & Runeson, P. (2011). *Usage of Open Source in Commercial Software Product Development – Findings from a Focus Group Meeting*. https://doi.org/10.1007/978-3-642-21843-9_13

Höst, M., & Runeson, P. (2007). Checklists for Software Engineering Case Study Research. *ESEM*, 479–481.

Kemp, R. (2010). Towards Free/Libre Open Source Software ("FLOSS") Governance in the Organisation. *International Free and Open Source Software Law Review*, *1*(2). https://doi.org/10.5033/ifosslr.v1i2.19

Koltun, P. (2011). *FOSS Compliance Practices for Supplied Software*.

Link, C. (2010). *Patterns for the commercial use of Open Source : Legal and licensing aspects*. https://doi.org/10.1145/2328909.2328918

Mutkoski, S. A. (2004). *Seven Steps to Addressing Open Source Issues in Software Development*.

Protecode. (2012). *Avoiding Enemies and Making Friends Using Embedded Open Source Software*.

Stewart, K. J., & Gosain, S. (2006). The Impact of Ideology on Effectiveness in Open Source Software Development Teams. *MIS Quarterly*, *30*(2), 291–314.

The Linux Foundation. (2012a). *A Template for Approval Request Form For The Use of Free and Open Source Software*.

The Linux Foundation. (2012b). *Linux Foundation Compliance Program : Generic FOSS Policy*.

Webster, J., & Watson, R. T. (2002). Analzying the Past to Prepare for the Future: Writing a Literature Review. *MIS Quarterly*, *26*(2), 13–23.

What is Free Software? (n.d.). Retrieved from http://www.gnu.org/philosophy/free-sw.html

Yin, R. K. (2013). *Case study research: Design and methods*. Sage publications.