Industry Best Practices for Open Source Governance and Component Reuse

Nikolay Harutyunyan
Computer Science Department
Friedrich-Alexander University Erlangen Nürnberg
Erlangen, Germany
nikolay.harutyunyan@fau.de

Dirk Riehle
Computer Science Department
Friedrich-Alexander University Erlangen Nürnberg
Erlangen, Germany
dirk@riehle.org

ABSTRACT

Corporate use of open source in software products is on the rise. While this brings a number of technological and business benefits to companies, it also comes with potential legal and financial risks caused by license non-compliance and ungoverned use of open source components. Companies address these threats with free/libre and open source software (FLOSS) governance - internal guidelines and processes for using open source components in products. An essential aspect of FLOSS governance is component reuse and component repository, which enable efficient governance for the previously used components by the company's developers. In our study, we aimed to identify the current industry best practices for FLOSS governance and component reuse. We conducted 15 expert interviews in companies with high governance maturity, analyzed these interviews and derived 19 best practices cast in the pattern format of context-problem-solution. The format was inspired by design patterns and enables higher applicability of our research results by practitioners. The 19 best practices form a handbook on FLOSS governance and component reuse that also includes workflows connecting the individual practices into process templates.

KEYWORDS

Best Practice, Commercial Use of Open Source, Component Repository, Component Reuse, FLOSS, Introduction of FLOSS in Companies, Open Source Software, Open Source Components, Open Source Governance, Pattern, Pattern Language

ACM Reference format:

Nikolay Harutyunyan and Dirk Riehle. 2019. Industry Best Practices for Open Source Governance and Component Reuse. In *Proceedings of EuroPLoP 2019. ACM, New York, NY, USA, 14 pages.* https://doi.org/10.1145/3361149.3361170

1 Introduction

Companies are using more and more open source components in their products. Such use can be beneficial, but also carries certain risks if a company has no guidelines. The inappropriate use of open source components can result in legal problems due to license non compliance, operational, financial and intellectual property issues resulting in litigation, cease and desist claims or product recalls [22, 24, 26]. To address and mitigate these risks companies must govern their FLOSS use through open source governance processes and guidelines.

We define FLOSS governance as the set of processes, best practices, and tools employed by companies to use FLOSS components as parts of their commercial products while minimizing their risks and maximizing their benefit from such use [10]. In the context of this paper, the definition of FLOSS governance should not be confused with other definitions that cover the governance of open source communities or projects. Namely, in the context of open source communities, governance refers to the processes by which decisions are made within the community [33]. In our definition, we focus solely on how companies govern their use of open source components internally and not on how companies contribute or lead open source communities. We limited the scope of this paper to the commercial use of open source components only, intentionally excluding governance considerations of companies contributing to or leading open source communities or projects. This focus enabled us to build and present a detailed study covering multiple aspects of component reuse and repository, which is a topic of interest for both FLOSS researchers and practitioners.

Different companies have different levels of maturity when it comes to corporate open source governance. The more mature the company is, the more aware it is of various governance issues and topics such as governance management, open source program office, license compliance, component search, component selection, component approval, component integration, component repository and reuse, software product model, supply chain management, communication, capabilities and more. In this paper we focus on companies that are aware of the need for governance, and already got started with establishing FLOSS governance processes and practices, but have not covered the whole space yet. Our scope is limited to the governance aspects of open source component reuse and component repository.

Research question: How should companies reuse open source components and maintain a component

repository in the context of open source governance based on existent industry best practices?

The research method we employed is a qualitative survey [13]. Our paper presents an interview-based qualitative survey exploring the open source governance introduction experiences of 15 software development companies that use open source components in their products, reuse these components and have component repositories for such components. We performed data gathering and analysis using formal semi-structured interviews. We interviewed FLOSS governance and compliance experts from 15 diverse companies chosen through theoretical sampling of more than 140 companies.

The contribution of our paper is a theory of industry best practices for component reuse and component repository. Our research results are presented in 19 best practices we discuss in this paper. We cast the individual best practices in the format of **context-problem-solution patterns**. This format is inspired by design patterns and has been previously applied to present academic research [25], as well as to study various aspects of the commercial open source use, such as the benefits of using open source software [31], leading open source communities [31], and legal considerations [16]. This pattern format is actionable and highly practice-relevant, as it enables industry to apply the findings of our research by adapting and applying the best practices we identified in their companies. The latter was a priority of ours, as we wanted to increase the potential impact of our work, as the target audience of our paper are companies that are currently using open source components in a grass-roots manner, without formal and systematic consideration of open source license compliance or efficient components reuse practices. When combined, patterns form process templates or workflows that guide the application of the identified best practices. We present two examples of such workflows in Figure 1 and Figure 2.

The resulting best practices and process templates form a practical handbook for FLOSS governance and component reuse. We are currently working on a handbook that goes beyond component reuse encompassing other aspects of open source governance and compliance, such as getting started with governance in companies, general management and open source review boards, approval incoming open source components, supply chain management, license tracking and compliance etc. Other researchers have studied some of these aspects, such as open source review board and liaison [9, 20], audit and due diligence [16, 20], tracking license changes [20]. However, this paper focuses solely on the part of our handbook related to open source component reuse. Other aspects of open source governance and compliance are out of scope for this paper.

2 Related Work

We set our research scope and that of the related work review to the commercial use of FLOSS components in products, FLOSS governance and open source component reuse. We employed snowballing as a search approach for literature research. We identified literature on tracking and reuse of FLOSS components [11, 21, 29].

With the growing availability of high-quality FLOSS components, software developers increasingly use FLOSS components in commercial products, which among other things enables faster development and lower costs as illustrated in the BOOTSTRAP pattern by Weiss [31]. On the other hand, the commercial use of open source comes with certain challenges [28] including dependency on the community of an open source project [4, 5, 15, 18], complex licensing [1, 12, 17], low quality documentation [2, 3, 17, 18] and other issues. The various issues and risks of using open source software commercially can be mitigated through open source governance process and practices. Open source governance addresses, among other issues, license compliance management [7] and related tooling [10, 14].

FLOSS governance policies in many companies require developers to track and document their open source use [11, 21]. This enables the well-structured management and reuse of FLOSS components that have been added into product software. Umarji et al. [30] suggest using FLOSS governance tools to create and maintain repositories of reusable FLOSS components. Our findings confirm this in the best practice patterns *OSGOV-COMREU-8-10, 12.* Other best practices focus on component reuse, integration with bill-of-materials [19, 27], maintenance of FLOSS component metadata in product architecture models [23], etc. Our theory confirms and captures these industry practices.

Ruffin & Ebert [26] talk about possible risks and benefits of using open source software. Besides advantages like saving time and improving security, they point out that companies should be vigilant about the open source components used in their products, preventing possible copyright infringement of third parties and their intellectual-property rights. They also talk about several actions that can be undertaken to mitigate legal exposure, such as governing the use of open source components through well documented processes. Once the company's developers go through this process, they can reuse the once used components and check them into a component repository. We confirmed their findings and identified best practices that help avoid potential risks of the ungoverned use of open source components, as well as develop efficiently through component reuse. The best practices *OSGOV-COMREU-11*, *OSGOV-COMREU-18* confirm these findings from related literature.

3 Related Work

We defined the following research question and subquestions for our study:

RQ1: How should companies using open source software in their products reuse open source components systematically, efficiently, in a legally compliant manner?

RQ1.1: How should companies guide their open source governance and component reuse?

RQ1.2: How should companies operationalize their open source governance and component reuse?

To answer these research questions, we conducted a qualitative survey using interviews with industry experts to collect data [6, 13]. We set objectives to collect information, design and plan the study, conduct a theoretical sampling, and choose expert interviews as our main source of data. We then prepared the interview questions that covered different pre-defined aspects or topics of open source governance and component reuse. Using semi-structured interviews as our survey instrument, we conducted the interviews in an iterative manner adjusting the questions after each iteration, yet keeping the core topics of the questions intact. We then transcribed and processed the interviews to prepare for data analysis. To analyze survey data, we employed qualitative data analysis (QDA) aided by MaxQDA (a QDA tool) in order to ensure the systematic analysis of the data and the traceability of our theory to the data. Finally, we are reporting our findings as a theory of industry best practices in this paper. A best practice is a method reflecting the state-of-the-art as applicable in a particular context [25]. This paper presents the best practices we developed in the results section.

We chose 14 companies and one open source foundation sampled from our professional network of about 140 companies and foundations with advanced FLOSS governance practices. The companies in our sample are experienced in FLOSS governance and compliance. We conducted polar theoretical sampling to cover a diverse and representative set of companies, which resulted in a sample with highly varying characteristics [6, 13]. To ensure a diverse set of sources we classified the companies and foundations in our professional network:

- by business domain
- by size (based on the revenue and number of employees)
- by type of customers
- by business models
- by market position
- by level of maturity.

We then chose 10 potential entities, contacted them and interviewed experts from the ones that were willing to take part in our study. We started our analysis following the first round of interviews, after which we added more companies to our sample and interviewed more experts to address the gaps from the first round of expert interviews. In total we conducted five rounds of sampling in order to achieve theoretical saturation. Our interview questions focused on various aspects of open source governance in companies, including but not limited to the reuse of open source components.

The list of companies and some of their characteristics are presented in Table 1. Company names are anonymized per their request.

Table 1. Theoretical sample of companies

Company	**Company domain**	**By size**	**By type of customer**
Company 1	Consulting	Medium	Enterprise
Company 2	Automotive	Small	Enterprise
Company 3	Automotive	Large	Enterprise
Company 4	Enterprise Software	Medium	Enterprise, retail
Company 5	Enterprise Software	Medium	Enterprise, retail
Company 6	Enterprise Software	Large	Enterprise, retail
Company 7	Enterprise Software	Medium	Enterprise, retail
Company 8	FLOSS Foundation	Small	Enterprise, retail
Company 9	Hardware and Software	Large	Enterprise
Company 10	Legal	Large	Enterprise, government
Company 11	Enterprise Software	Medium	Enterprise
Company 12	Consulting, Enterprise Software	Large	Enterprise
Company 13	Hardware and Software	Large	Enterprise, retail, government
Company 14	Enterprise Software	Small	Enterprise
Company 15	Enterprise Software	Large	Enterprise

4 Research Results

Our study resulted in a set of industry best practices presented using context-problem-solution patterns that are interconnected forming an open source governance handbook on component reuse and component repository.

A best practice description follows a pattern format and consists at least of the three sections: context, problem, and solution. There may also be references, examples, and other sections. A best practice description abstracts from examples in multiple sources; it aims to represent a general truth applicable widely within its context.

Best practices link to each other. Ideally, a user of the handbook starts with the first best practice and from there works his or her way through the handbook, applying one practice after another. This sequence of applications represents the establishment of the desired processes at the company. The handbook structure and the section hierarchy allow a user of the handbook to orient themselves and jump into the middle of where they last left off.

Not all best practices need to be applied in order to achieve the company’s goals. In fact, many times the best practice application is more like a decision tree where an analysis of the context and problems at hand indicate which of several possible best practices is to be used next. Ideally, all links between best practices are forward links, implying that if a best practice A precedes another best practice B in the handbook, it should be applied before it.

Our theory of industry best practices for open source governance and component reuse focused on:

- definition of a component reuse policy
- operationalization of the component reuse policy in a component reuse process
- creation and maintenance of a searchable component repository with a single well-defined location

- guidelines for auditing and using the component repository
- integration of component reuse processes with other aspects of open source governance.

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

OSGOV-COMREU-1. Establish component reuse policy
OSGOV-COMREU-2. Communicate component reuse policy
OSGOV-COMREU-3. Adjust and improve component reuse policy
OSGOV-COMREU-4. Designate a role of responsibility for the component repository, in multiple places in the company
OSGOV-COMREU-5. Establish component reuse process
OSGOV-COMREU-6. Communicate component reuse process
OSGOV-COMREU-7. Implement component reuse process
OSGOV-COMREU-8. Create component repository
OSGOV-COMREU-9. Update component repository
OSGOV-COMREU-10. Maintain component repository
OSGOV-COMREU-11. Audit component repository
OSGOV-COMREU-12. Use tools to create, update and maintain component repository
OSGOV-COMREU-13. Provide component repository a single well-defined location
OSGOV-COMREU-14. Track prior approval data for reuse
OSGOV-COMREU-15. Provide all relevant metadata for component
OSGOV-COMREU-16. Search component repository for reusable components
OSGOV-COMREU-17. Contact OSPO for details on a repository entry
OSGOV-COMREU-18. Add security check information to component repository
OSGOV-COMREU-19. Link BOM and component repository.

The above mentioned best practices are presented in full detail in the following subsections.

OSGOV-COMREU-1. Establish component reuse policy

Name	Establish component reuse policy
Actor	OSPO (Open Source Program Office)
Context	After defining goals of governance and establishing an open source program, you defined roles, responsibilities, and policies to address various aspects of open source governance in an abstract manner.
Problem	Without a component reuse policy, developers must file new component approval requests for every new component without an ability to reuse the open source components that have already been approved within the company. Especially in large companies, the redundancies in the defined component approval process can be very costly and inefficient. How can this problem be solved in a systematic manner?
Solution	**You need to define the specific policy for open source component reuse within the company. Having such a policy will reduce the ungoverned use of open source components by developers, will enable efficient reuse of open source components across company products and projects, and will help document the current use and reuse of open source components.** In parallel to → *establishing the component reuse process*, establish component reuse policy that specifies the company's approach to reusing open source components that have already been approved by the Open Source Program Office. The policy often has two logical parts: intention of the policy explaining the Open Source Program Office's motivation for having this policy, and guidelines & best practices that are later operationalized through the → *established component reuse process*. In particular, the aspects covered by the open source component reuse policy include but are not limited to: ● company goals and metrics for component reuse ● principles for reusing open source components ● recommendations for automating component reuse though tools ● rules and guidelines for reusing open source components ● component reuse integration with other aspects of open source governance, including component approval, bill-of-materials, component tracking, and license compliance ● recommended best practices for reusing open source components. Select industry best practices for reusing open source components are presented in this

	handbook, such as: ➔ *Designate a role of responsibility for the component repository, in multiple places in the company* ➔ *Provide a component repository in a single well-defined location* ➔ *Use tools to create, update and maintain a component repository* ➔ *Track prior approval data for reuse* ➔ *Add security check information to the component repository* ➔ *Link the BOM and the component repository* An open source component reuse policy helps the Open Source Program Office define a consistent approach to the issue that can be systematically documented, implemented and communicated across the organization. It should evolve and can be modified by the Open Source Program Office, when necessary. An open source component reuse policy should be easy to read and to the point with appropriate references to other parts of this handbook. As a starting point the policy should consider the component repository pre-established during component approval. Once a component approval decision is made, Open Source Program Office adds the decision to component repository. This repository should be used to ⟶ *create a component repository* for component reuse that includes additional data for each open source component, such as: ● when, where and how an approved component was reused ● whether it was additionally audited for compliance ● whether it was additionally checked for export restrictions ● whether it was additionally checked for security or for quality assurance etc. The policy is then operationalized through the ⟶ *established component reuse process* that defines the steps that developers need to follow in order to efficiently reuse an approved open source component in their products or projects. Once the policy is established, it is necessary to ⟶ *communicate component reuse policy* and to ⟶ *adjust and improve component reuse policy*.

OSGOV-COMREU-2. Communicate component reuse policy

Name	Communicate component reuse policy
Actor	OSPO (Open Source Program Office)
Context	After ⟶ *establishing a component reuse policy*, it is necessary to make the component reuse policy accessible to the employees, so they can follow it and ask clarification questions to the OSPO.
Problem	What are the best channels to communicate the component reuse policy?
Solution	**Communicate the component reuse policy using the regular channels that OSPO uses for open source governance and compliance related internal communication.** Use the communication channels for both spreading the component reuse policy designed by the OSPO and for ⟶ *adjusting and improving the component reuse policy* based on the feedback from the employees following the policy. The main recipients of OSPO's communication should be the employees in different development teams that occupy the ⟶ *designated role of responsibility for the component repository*. For some parts of the component reuse policy, these employees should forward the specifics or changes of the policy to individual developers. At the same time, these designated employees are responsible for collecting the commonly asked developer questions and forwarding them to the OSPO, as well as collecting and sharing OSPO's answers with colleagues.

OSGOV-COMREU-3. Adjust and improve component reuse policy

Name	Adjust and improve component reuse policy
Actor	OSPO (Open Source Program Office)
Context	You ⟶ *established* and ⟶ *communicated component reuse policy*. You are now getting clarification requests, questions and feedback on the policy from the employees.
Problem	What's the best way to address employee feedback regarding component reuse policy?
Solution	Iteratively collect and analyze the questions and requests you are getting. Identify the common misunderstandings and use them to adjust and improve the policy. Instruct the ⟶ *designated component repository responsible roles across the company* to regularly post updates, clarifications, and FAQs to the affected employees via internal communication channels used to ⟶ *communicate component reuse policy*. It's important to update and maintain the policy by receiving feedback from the development teams to improve the processes. It is also essential to record violations to understand why certain development teams violate the process to minimize the issues in the future.

OSGOV-COMREU-4. Designate a role of responsibility for the component repository, in multiple places in the company

Name	Designate a role of responsibility for the component repository, in multiple places in the company
Actor	OSPO (Open Source Program Office)
Context	Open source components are used in virtually all software products. For most companies this means that many different projects in different divisions of the company face the same issues of component reuse.
Problem	How can you ensure and manage the efficient component reuse across your company with optimal resource use?
Solution	The OSPO is responsible for open source governance in general and for component reuse in particular. However, with an issue like component reuse and repository a centralized body like the OSPO needs local support in different divisions and teams of the company. OSPO's role is to: ➔ *Establish component reuse policy* ➔ *Communicate component reuse policy* ➔ *Adjust and improve component reuse policy* ➔ *Establish component reuse process* ➔ *Communicating component reuse process* However, OSPO does not have the resources to locally ⟶ *implement component reuse process* in every project and division of the company. For efficient component reuse, you need a centralized and unified approach and organization. **OSPO should create a role of responsibility for the component repository which can be delegated to employees within different projects in the company. Depending on the company, recommended candidates for such a role include:** ● **(software) project managers** ● **(software) product managers** ● **technical product managers** ● **senior developers.** People responsible for component repositories should spend only a small part of their work time in component reuse support. However, this role can be combined with other similar responsibilities in other areas of open source governance. Component repository responsibles support their teams to: ➔ *Update the component repository* ➔ *Maintain the component repository* ➔ *Track prior approval data for reuse* ➔ *Provide all relevant metadata for a component* ➔ *Search the component repository for reusable components* ➔ *Contact the OSPO for details on a repository entry*.

OSGOV-COMREU-5. Establish component reuse process

Name	Establish component reuse process
Actor	OSPO (Open Source Program Office)
Context	After ⟶ *establishing a component reuse policy* in accordance with the company's ⟶ *defined goals of governance*, you must now define how exactly should the employees reuse open source components in the company's products.
Problem	You ⟶ *established a component reuse policy* and ⟶ *communicated a component reuse policy*. However, the reuse policy is too broad to apply operationally. It focuses on the company's principles for reusing open source components, but leaves out the operational aspects of component reuse. How should you operationalize the component reuse policy, while making sure it is widely accepted and used?
Solution	Specify the company's approach to reusing open source components that have already been approved by the Open Source Program Office. As open source component reuse is virtually inevitable and necessary for efficient development, **a process will ensure that the current local solutions or workarounds for component reuse are replaced by a centrally defined and unified process.** Such a process has a number of benefits including, but not limited to: ● point of reference for new developers or managers who need to reuse an open source component ● a centralized database or component repository with all the used open source component across the organization ● consistent and up-to-date metadata for the used open source components ● an easy search of the open source components in use and metadata of this use, including component approval decisions, information on where these components have been used, and information on previous component owners. The component reuse process should follow these principles: - be clearly defined - be easy to follow without constant guidance by the OSPO (OSPO should handle the exceptions on a case by case basis) - be scalable and replicable - be assisted with tools that automate and/or ensure compliance with the process. First, you need to establish preparatory steps for component reuse process. For this, you need to: ➔ *Establish a component reuse policy*

	➔ *Communicate the component reuse policy* ➔ *Adjust and improve the component reuse policy* ➔ *Designate a role of responsibility for the component repository, in multiple places in the company* Second, you need to establish a process for the main artefact of component reuse - the component repository, which is the company-internal database with all the used open source components, their component approval decisions and details, their metadata, and reuse information. For this, you need to: ➔ *Create a component repository* ➔ *Update the component repository* ➔ *Maintain the component repository* ➔ *Audit the component repository* Third, you need to establish the essential process for the component reuse. For this, you need to: ➔ *Track prior approval data for reuse* ➔ *Provide all relevant metadata for a component* ➔ *Search the component repository for reusable components* ➔ *Contact the OSPO for details on a repository entry* Depending on your specific needs you should modify the essential process to include more steps that would guide the employees in reusing open source components.

OSGOV-COMREU-6. Communicate component reuse process

Name	Communicate component reuse process
Actor	OSPO (Open Source Program Office)
Context	After → *establishing a component reuse process*, you must communicate it clearly with the → *employees with the designated role of responsibility for the component repository* and other affected employees who will be using the process.
Problem	The component reuse process must be communicated to the affected employees across the company. The process can be complex and not all parts of the process are relevant for everyone. How should you communicate the component reuse process before → *implementing the component reuse process*?
Solution	**You should present an overview of the component reuse process to the affected employees, and point each employee group / role to the best practice patterns relevant only to them, while guiding them and serving as a central hub for discussions about the overarching issues.** Your direct communication should be with the → *employees with the designated role of responsibility for the component repository*. After that, it's recommended that the people responsible for the component repository meet their teams to process and to clarify the steps, focusing on: - preparatory steps for the component reuse process - the process for the main artefact of component reuse - the component repository - the essential process for the component reuse. For efficient communication, use the regular channels that the OSPO uses for open source governance and compliance related internal communication.

OSGOV-COMREU-7. Implement component reuse process

Name	Implement component reuse process
Actor	OSPO (Open Source Program Office), Component repository responsibles
Context	After → *establishing a component reuse process* and → *communicating the component reuse process*, you must implement the component reuse process across the company.
Problem	Implementing a large-scale process across the company has its challenges. How should you implement an efficient component reuse process?
Solution	**Gradually implement the component reuse process. Start by preparing your organization for governed component reuse. In this phase, apply the following best practices, if they have not been implemented yet:** ➔ ***Establish a component reuse policy*** ➔ ***Communicate the component reuse policy*** ➔ ***Adjust and improve the component reuse policy*** ➔ ***Designate a role of responsibility for the component repository, in multiple places in the company*** The OSPO then → *creates a component repository*, which is updated and maintained by people responsible for the component repository and developers. The repository is audited by the OSPO. In this phase, follow these best practices: ➔ *Create the component repository* ➔ *Update the component repository* ➔ *Maintain the component repository* ➔ *Audit the component repository* In the final step of process implementation, people responsible for the component repository should instruct individual developers to follow the essential process for component reuse. In this phase, follow these best practices:

	➔ *Track prior approval data for reuse* ➔ *Provide all relevant metadata for a component* ➔ *Search the component repository for reusable components* ➔ *Contact the OSPO for details on a repository entry* Throughout the implementation of the component reuse process, use industry best practices, including but not limited to: ➔ *Use tools to create, update and maintain a component repository* ➔ *Provide a component repository in a single well-defined location* ➔ *Add security check information to the component repository* ➔ *Link the BOM and the component repository* Finally, the OSPO should handle the exceptions that deviate from the implemented process on a case by case basis, while considering process optimization and continuous improvement.

OSGOV-COMREU-8. Create component repository

Name	Create component repository
Actor	OSPO (Open Source Program Office)
Context	When → *implementing a component reuse process*, one of the key tasks of the OSPO is to establish a database where open source components and their metadata can be stored. This database is then used by all the employees in the company, when needed.
Problem	How should you store and share the used open source components, their metadata and reuse information across the organization?
Solution	During open source component approval, the **OSPO needs to set up a component repository based on the documented current open source use from the getting started process. This component repository includes the relevant information about all open source component approval requests, including the data from the request checklist from the filed component approval requests. The repository includes information on all the requests with both approval and rejection decisions by the OSPO.** The repository should be open to the developers of the company, so they can consult it before making a new request, as the same license/use case pair may have already been assessed in the past. Such a situation enables easy component reuse, which is one of the key goals of the component repository. The repository needs to be searchable and easy to use for developers and for the OSPO. It should have a well-defined structure and be maintained by the OSPO. There are different approaches for designing component repositories. Depending on your needs you could store components, their approval/rejection decision, their metadata and reuse information in a database, a wiki-like system for (unstructured) metadata with background on previous decisions, a tool-integrated database, or an internally developed hybrid. At its core the component repository must store the following data based on the provided approval request templates, for each component: - name and ID of the developer - name and ID of the organizational unit - component approval checklist needed to file a component approval request: - Component ID and name - Component address / location - Product / Project ID and name - Product version - Open source license name - Multiple licenses (y/n) - Open source license version - Copyright holder - Linkage type to the rest of the (software) product (e.g. dynamic or static) - Use case (e.g. internal use only, to be directly distributed as part of the product, used to compile software to be distributed as part of the product) - component reuse information - Has the component (with its unchanged license and version) been used in the company before (can be automatically identified) - Same information as component approval checklist above (to identify any changes in use, e.g. a different version or a changed license) - Well-defined location in the component repository/database (for easy sharing internally) When creating the component repository, follow these best practices: ➔ *Use tools to create, update and maintain a component repository* ➔ *Provide a component repository in a single well-defined location* ➔ *Link the BOM and the component repository* Create a component repository with its use cases in mind. Make sure it is scalable and easy to use. After creating the repository, you should

	regularly: ➔ *Update the component repository* ➔ *Maintain the component repository* ➔ *Audit the component repository*.

OSGOV-COMREU-9. Update component repository

Name	Update component repository
Actor	People responsible for the component repository, developers
Context	After → *creating a component repository*, one needs to update it with the components that have been approved or rejected for use, as well as their metadata. The up-to-date repository is necessary for the component reuse process to work efficiently.
Problem	How should you update the component repository?
Solution	People responsible for the component repository must regularly update the component repository to ensure timely and efficient reuse of open source components within the company. The repository is mostly updated by people responsible for the component repository or by developers directly (e.g. after an open source component goes through component approval process), but the OSPO should regularly review the repository and its updates to ensure the overall consistency and completeness. To update the repository, people responsible for the component repository should: ➔ *Add component approval decisions to the component repository* ➔ *Provide all relevant metadata for a component* ➔ *Add security check information to the component repository*.

OSGOV-COMREU-10. Maintain component repository

Name	Maintain component repository
Actor	People responsible for the component repository, developers
Context	After the OSPO → *creates a component repository*, and people responsible for the component repository and developers → *update component repository regularly*, the component repository needs maintenance.
Problem	How should you maintain the component repository?
Solution	People responsible for the component repository and developers must maintain the component respiratory with the support of the OSPO, when needed. After the first version of the repository is created, people responsible for the component repository, developers and the OSPO will recognize some of the limitations of the first version, some bugs will be identified and some new requirements will arise. To address these challenges, you need to perform regular maintenance where you fix the identified issues and add new functionality to the repository (e.g., new search algorithms, better sharing functionalities). You might also need to adapt your component repository to existing or newly introduced governance tools and databases (e.g., integration with the product architecture model).

OSGOV-COMREU-11. Audit component repository

Name	Audit component repository
Actor	OSPO (Open Source Program Office)
Context	After the OSPO → *creates a component repository*, and people responsible for the component repository and developers → *update the component repository regularly*, the component repository needs to be audited.
Problem	How should you audit the component repository?
Solution	Just as the component repository needs to be → *maintained*, its content needs to be audited for consistency and completeness. The OSPO should perform regular audits of the component repository entries. These audits should ensure that open source components are properly documented when used and reused. They should also uncover any difficulties that developers have using the repository. If an inconsistency is identified, the OSPO needs to look into the issue with support from people responsible for the component repository and developers in a given project team. After the audit people responsible for the component repository and developers should → *update the component repository* to fix the identified issues.

OSGOV-COMREU-12. Use tools to create, update and maintain component repository

Name	Use tools to create, update and maintain component repository
Actor	OSPO (Open Source Program Office), people responsible for the component repository, developers
Context	In parallel to → *implementing the component reuse process*, you will recognize that some parts of the process can be automating for increased efficiency compared to the fully manual process.
Problem	How should you automate the component reuse process?
Solution	The component reuse process can be very

	laborious if done manually or using only basic tools, such as spreadsheets. For increased efficiency and scalability of your process, consider automating parts of it. Start by using tools to: ➔ *Create a component repository* ➔ *Update the component repository* ➔ *Maintain the component repository* When creating a component repository, use tools to store components, their approval/rejection decision, their metadata, and reuse information in a database, a wiki-like system for (unstructured) metadata with background on previous decisions, a tool-integrated database, or an internally developed hybrid. Make sure to employ the tools you are already using for Getting Started, Component Approval and other topics of open source governance. It is recommended to use open source governance tools integrated with common (standard) formats for open source component metadata (e.g., SPDX). The latter can be useful for supply chain management and outbound governance, and in particular for → *linking the BOM and the component repository*. To select the right tools, study your company's FLOSS governance needs, and scale of open source component usage and reuse. You should consider both open source tools and proprietary tools for open source governance.

OSGOV-COMREU-13. Provide component repository a single well-defined location

Name	Provide component repository a single well-defined location
Actor	OSPO (Open Source Program Office)
Context	Individual project teams in the company have their own local repositories for software components they are using. Such repositories are isolated from other project teams and from the rest of the company.
Problem	How can you ensure a company-wide open source component reuse that overcomes organizational silos?
Solution	When → *creating a component repository*, make sure to provide a single well-defined location where the developers and people responsible for the component repository should go to for all tasks related to component reuse and component repositories. Having a single well-defined location for the component repository encourages component reuse, enables ease of use, scalability, use of tools, and helps overcome the company's organizational silos that prevent sharing open source components.

OSGOV-COMREU-14. Track prior approval data for reuse

Name	Track prior approval data for reuse
Actor	OSPO (Open Source Program Office), developers
Context	During the component approval process, the OSPO → *adds component approval decisions to the component repository*. This helps the company mark what open source components have been used in the past.
Problem	How can the OSPO and developers know which approved open source components have been used more than once since their approval?
Solution	The OSPO and developers must track prior approval data for reuse. They must integrate the data on the → *recorded component approval decisions* into the newly created → *repository for component reuse*, where in addition to component approval data, developers also record and track component reuse data. You should track component data and reuse metadata meticulously, in order to have an up-to-date repository at all times. This enables developers to efficiently reuse the previously used open source components, and helps speed up license compliance if integrated with the component approval and reuse processes. For efficient tracking of open source components which have been used, → *audit the component repository* and make sure that the identified components match those recorded in the repository. If needed, fix the inconsistencies. It is also recommended to employ → *tools to create, update and maintain a component repository*.

OSGOV-COMREU-15. Provide all relevant metadata for component

Name	Provide all relevant metadata for component
Actor	Developers
Context	When the OSPO → *created a component repository*, it defined what component data and metadata must be stored there. To → *update the component repository*, employees must enter the defined component data and metadata into the repository.
Problem	How should employees enter the necessary data into the component repository?
Solution	Whenever a new open source component is used by a developer, it must be stored in the component repository. Developers must provide all relevant metadata for the used component(s). Storing this metadata enables easy reuse of the components by other developers in the company. It also enables → *searching the component repository for reusable components* using metadata

	filters or tags. The metadata you need to provide includes: - name and ID of the developer - name and ID of the organizational unit - component approval information - component approval checklist needed to → *file a component approval request* - component approval metadata from → *tracking prior approval data for reuse* - component reuse information - Has the component (with its unchanged license and version) been used in the company before (can be automatically identified) - Same information as component approval checklist above (to identify any changes in use, e.g. a different version or a changed license) - Well-defined and unique location in the component repository/database (for easy sharing internally) - other metadata (if needed) - security check data from - expert restriction data To add the relevant metadata into the component repository, use → *tools to create, update and maintain a component repository* in order to automate when possible.

OSGOV-COMREU-16. Search component repository for reusable components

Name	Search component repository for reusable components
Actor	Developers
Context	Developers are the main stakeholders of open source component reuse. During software development they normally reuse software components locally within their projects. While it might sound attractive to share and reuse software components across the whole organization, it can be challenging without the appropriate centralized infrastructure. Once the OSPO → *creates a component repository* and → *implements the component reuse process*, developers can start reusing open source component easily.
Problem	How should developers find the appropriate open source components to reuse in their projects?
Solution	When → *creating a component repository*, the OSPO made sure it is easily searchable by the developers and designated component repository responsible employees. Searching the repository before looking for other open source or proprietary components can save developer resources and time, increase productivity, and encourage internal collaboration. When → *searching for open source components* during software development, developers should consult the component repository to identify the previously used components that fulfill the functional requirements at hand. The scope of the search should include the whole company-wide repository and not be limited by organizational silos. Developers should be able to search using various filters that enable better usability. Component search should be supported by → *tools to create, update and maintain a component repository*. Searching the component repository should show → *all the relevant metadata for open source components* provided with the components entered into the repository. For easier search, you can also consider using tags, labels, and common naming conventions.

OSGOV-COMREU-17. Contact OSPO for details on a repository entry

Name	Contact OSPO for details on a repository entry
Actor	People responsible for the component repository, developers
Context	You can → *search for open source components* in the component repository to find out component data and metadata. However, some entries in the repository might need clarification or further details.
Problem	How can you clarify certain details on an open source component in the repository?
Solution	If a developer needs additional details about an open source component in the repository, they should contact the OSPO to get details on a repository entry. Developers can also contact the person responsible for the component repository in their team, who then contacts the OSPO to inquire for additional details. Such details might include notes or case by case decision documentation for exceptions from the component search, component selection, component approval, or component reuse processes. They might also store additional details about trademarks, patents or other relevant metadata that goes beyond the required → *relevant metadata to be provided for a component*. To contact the OSPO, use the regular channels that OSPO uses for open source governance and compliance related internal communication.

OSGOV-COMREU-18. Add security check information to component repository

Name	Add security check information to component repository
Actor	OSPO (Open Source Program Office)
Context	After ⟶ *creating a component repository*, you started ⟶ *updating the component repository*.
Problem	What should you add to the repository beyond the component data and metadata?
Solution	You should use the open source component repository beyond the needs of component reuse. One such use can be storing the security check information for each component. This will help you not only when reusing the component, but also during the release review. It is recommended to add security check information to the repository. If you have an existing database for security checks, you should integrate this database with the component repository. If you are using specialized tools for scanning the security vulnerabilities in open source components, you should integrate the output of these tools into the component repository. Make sure to regularly update the security check information in order not to miss newly discovered vulnerabilities.

OSGOV-COMREU-19. Link BOM and component repository

Name	Link BOM and component repository
Actor	OSPO (Open Source Program Office)
Context	After ⟶ *creating a component repository*, you started ⟶ *updating the component repository* for potential reuse of the open source components. At the same time, you need to document the bills-of-materials (BOM) of your products, including the open source components you used in these products.
Problem	How can you use the component repository for better supply chain management and BOM documentation?
Solution	You should use the open source component repository beyond the needs of component reuse. One such use can be automating certain aspects of software supply chain management by linking the component repository and your current BOM documentation. BOM documentation shows which open source component is used in which product, and all the relevant metadata, such as the version of the component, the copyright notice, etc. It is recommended to use a common standard for BOM documentation (e.g., SPDX). This will enable the easy integration with the component repository, and the automated tracking of the open source component interdependencies in the software products.

5 Conclusion

After defining and detailing the derived best practices, we concluded our work by suggesting process templates, workflows consisting of individual best practices that would help practitioners apply our handbook on FLOSS governance and component reuse.

We found that companies intending to reuse open source components in the scope of FLOSS governance should start by defining and communicating a component reuse process, which is then operationalized in a component reuse process. Next, a company needs to create and update a component repository in line with company's open source governance policy. While the component repository is audited and maintained, developers should use it to track and obtain relevant information and metadata on previously used components, as well as collaborate with the open source program office for inquiries on specific components and their history. Finally, open source component repository needs to be integrated with other artifacts and tools of open source governance, such as license and security compliance tooling, and product bill-of-materials.

Figure 1 and Figure 2 illustrate two examples of process templates for open source component reuse.

6 Research Limitations

To address the limitations of our study, we follow [8] in assessing the trustworthiness of our research through the following quality criteria:

Credibility. Credibility is the degree to which we can establish confidence in the truth of our findings in the context of the inquiry. To ensure credibility during data collection we conducted our interviews iteratively, adjusting our semi-structured interview questions based on the company's context and on our experience with earlier interviews. We also conducted peer debriefing regarding our findings.

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

ACKNOWLEDGMENTS

This research was funded by BMBF's (Federal Ministry of Education and Research) Software Campus 2.0 project (OSGOV, 01IS17045-17570). We would like to thank our colleagues and the anonymous reviewers for their feedback. We would also like to thank our industry partners that provided their valuable time and expertise for this research project.

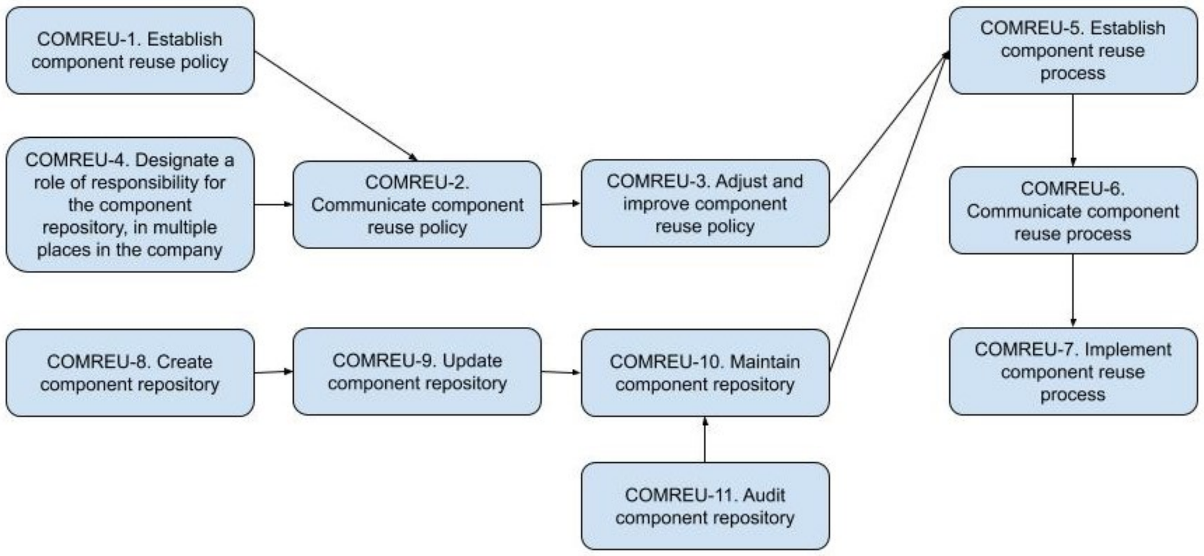
COMREU-1. Establish component reuse policy
COMREU-4. Designate a role of responsibility for the component repository, in multiple places in the company
COMREU-2. Communicate component reuse policy
COMREU-3. Adjust and improve component reuse policy
COMREU-5. Establish component reuse process
COMREU-6. Communicate component reuse process
COMREU-7. Implement component reuse process
COMREU-8. Create component repository
COMREU-9. Update component repository
COMREU-10. Maintain component repository
COMREU-11. Audit component repository

Figure 1. An example of a workflow diagram for FLOSS Governance and Component Reuse - Process Template A

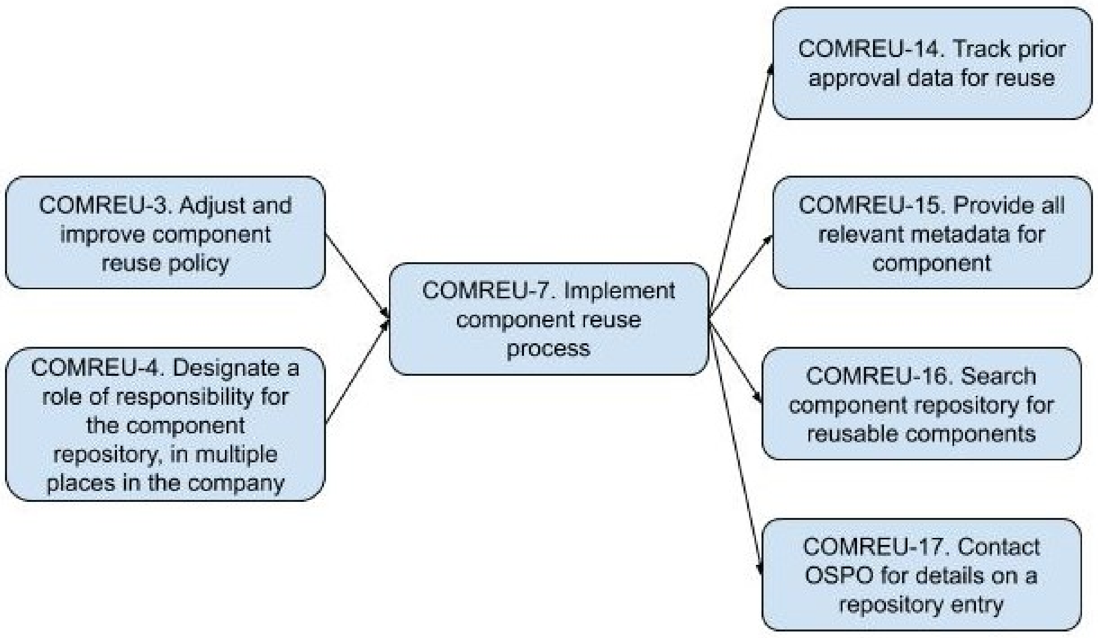
COMREU-3. Adjust and improve component reuse policy
COMREU-4. Designate a role of responsibility for the component repository, in multiple places in the company
COMREU-7. Implement component reuse process
COMREU-14. Track prior approval data for reuse
COMREU-15. Provide all relevant metadata for component
COMREU-16. Search component repository for reusable components
COMREU-17. Contact OSPO for details on a repository entry

Figure 2. Another example of a workflow diagram for FLOSS Governance and Component Reuse - Process Template B

REFERENCES

[1] Agerfalk, P. J., Deverell, A., Fitzgerald, B., & Morgan, L.: Assessing the role of open source software in the European secondary software sector: a voice from industry (2005)

[2] Akkanen, J., Demeter, H., Eppel, T., Ivánfi, Z., Nurminen, J. K., & Stenman, P.: Reusing an open source application—practical experiences with a mobile CRM pilot. In IFIP International Conference on Open Source Systems (pp. 217-222). Springer, Boston, MA (2007)

[3] Ayala, C., Hauge, Ø., Conradi, R., Franch, X., Li, J., & Velle, K. S.: Challenges of the open source component marketplace in the industry. In IFIP International Conference on Open Source Systems (pp. 213-224). Springer, Berlin, Heidelberg (2009)

[4] Chen, W., Li, J., Ma, J., Conradi, R., Ji, J., & Liu, C.: An empirical study on software development with open source components in the chinese software industry. Software Process: Improvement and Practice, 13(1), 89-100 (2008)

[5] Conlon, P., & Carew, P.: A risk driven framework for open source information systems development (2005)

[6] Fink, A.: Analysis of qualitative surveys. In: The survey handbook, 61–78. SAGE Publications, California (2003)

[7] Gangadharan, G. R., D'andrea, V., De Paoli, S., & Weiss, M.: Managing license compliance in free and open source software development. Information Systems Frontiers, 14(2), 143-154 (2012)

[8] Guba, E. G.: Criteria for assessing the trustworthiness of naturalistic inquiries. In: Educational Technology Research and Development, 29(2), 75–91 (1981)

[9] Gurbani, V. K., Garvert, A., & Herbsleb, J. D.: Managing a corporate open source software asset. Communications of the ACM, 53(2), 155-159 (2010)

[10] Harutyunyan, N., Bauer, A., Riehle, D.: Understanding Industry Requirements for FLOSS Governance Tools. In: IFIP International Conference on Open Source Systems, 151-167 (2018)

[11] Helmreich, M.: Best practices of adopting open source software in closed source software products. In: (Diplomarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg) (2011)

[12] Jaaksi, A.: Experiences on product development with open source software. In IFIP International Conference on Open Source Systems (pp. 85-96). Springer, Boston, MA (2007)

[13] Jansen, H.: The logic of qualitative survey research and its position in the field of social research methods. In: Forum Qualitative Sozialforschung/Forum: Qualitative Social Research, 11(2) (2010)

[14] Kapitsaki, G. M., Tselikas, N. D., & Foukarakis, I. E.: An insight into license tools for open source software systems. Journal of Systems and Software, 102, 72-87 (2015)

[15] Krivoruchko, J.: The use of open source software in enterprise distributed computing environments. In IFIP International Conference on Open Source Systems. Springer, Boston, MA. 277-282 (2007)

[16] Link, C.: Patterns for the commercial use of open source: legal and licensing aspects. In Proceedings of the 15th European Conference on Pattern Languages of Programs (p. 7). ACM (2010)

[17] Madanmohan, T. R.: Notice of violation of IEEE publication principles open source reuse in commercial firms. Ieee Software, 21(6), 62-69 (2004)

[18] Merilinna, J., & Matinlassi, M.: State of the art and practice of open source component integration. In 32nd EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO'06) (pp. 170-177). IEEE (2006)

[19] OpenChain Specification: https://www.openchainproject.org/spec. (2018)

[20] Peters, S.: Best Practices for Creating an Open Source Policy. In: OpenLogic (2009)

[21] Popp, K. M.: Best Practices for Commercial Use of Open Source Software. In: Business models, processes and tools for managing open source software. BoD–Books on Demand (2015)

[22] Radcliffe, M., Odence, P.: The 2017 Open Source Year in Review. In: Black Duck Software, DLA Piper. (self-published presentation) (2017)

[23] Riehle, D., Harutyunyan, N.: License Clearance in Software Product Governance. In: NII Shonan. (2017).

[24] Riehle, D., Lempetzeder, B.: Erfolgsmethoden der Open-Source-Governance und Compliance. In: Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU). (2014)

[25] Riehle, D.: Lessons Learned from Using Design Patterns in Industry Projects. In: Transactions on Pattern Languages of Programming II, LNCS 6510. Springer-Verlag, 1-15 (2011)

[26] Ruffin, C., Ebert, C.: Using open source software in product development: A primer. In: IEEE Software, 21(1), 82-86 (2004)

[27] Software Package Data Exchange (SPDX). https://spdx.org/ (2018)

[28] Stol, K. J., & Ali Babar, M.: Challenges in using open source software in product development: a review of the literature. International Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development, 17-22, ACM (2010)

[29] Tools for Managing Open Source Programs 2018. https://www.linuxfoundation.org/tools-managing-open-source-programs/ (2018)

[30] Umarji, M., Sim, S. E., Lopes, C.: Archetypal internet-scale source code searching. In: IFIP International Conference on Open Source Systems (pp. 257-263). Springer, Boston, MA. (2008)

[31] Weiss, M.: Profiting from open source. In Proceedings of the 15th European Conference on Pattern Languages of Programs (p. 5). ACM (2010)

[32] Weiss, M.: Profiting even more from open source. In Proceedings of the 16th European Conference on Pattern Languages of Programs (p. 1). ACM (2012)

[33] West, J. & O'Mahoney, S.: The Role of Participation Architecture in Growing Sponsored Open Source Communities, Industry and Innovation, 15(2), 145-168 (2008)