FOSS Governance Fundamentals

HP Part Number: 5992-4059
Published: January 2008
Edition: 1.0

Table of Contents

List of Figures

Introduction

Executive Summary

The acceptance and adoption of free and open source software (FOSS) is widespread and expanding rapidly across many industries. The use of FOSS is an attractive asset for many parts of an organization including development, IT operations, and IT strategy. For an organization to successfully deploy or develop FOSS, a system of proper IT governance and management is critical.

The use of FOSS in an organization carries a unique set of complexities. Techniques commonly used for the governance of traditional software do not adequately address open source issues. To develop a proper FOSS governance system, an organization needs an understanding and appreciation of the specific requirements associated with using open source software throughout its entire lifecycle. A comprehensive FOSS governance system must include processes to track its use and ensure compliance with a growing set of issues, such as acquisition, licensing, support, and distribution.

This document is intended as a primer on FOSS governance fundamentals and covers the following topics:

- Understanding IT governance
- Understanding open source
- Identifying the benefits and potential risks of using FOSS
- Understanding open source policies and guidelines
- Developing FOSS governance strategies
- Managing the use and proliferation of FOSS
- Using compliance tools

Intended Audience

This document is intended for a cross-organizational audience, and includes software development, auditing, legal, procurement, operational risk management, technology strategy, and line of business departments.

Understanding IT Governance

IT governance can be defined as a set of processes to effectively manage all IT assets, functions, and processes which support business goals and the fusion of business and IT. An enforceable program of IT governance is the key to gaining more value from open source and protecting the interests of your organization.

It is important to understand the context in which FOSS governance fits into your organization's overall IT governance system. Understanding and targeting open source issues within your organization is the first of many steps in developing and implementing a FOSS governance system. It is crucial to have a full understanding of the impact that using open source has in your organization. The following section identifies the areas within an organization's IT governance structure that are likely to be impacted by the use of FOSS.

Starting a FOSS Governance Program

The ways in which your organization responds to FOSS governance depends on the nature of the business. As you move forward into the challenges and opportunities associated with using FOSS, it is important to consider the areas impacted by IT governance in order to determine the interactions with open source governance. The objective is to identify ways to implement or improve an existing FOSS governance program that is suitable for your organization.

Scope of FOSS Governance

FOSS use within an organization can range from simple deployment to more complex applications in development projects. Keeping in mind the intended use of FOSS throughout an organization, it is important to determine which areas of the organization are impacted and to what extent. To properly address FOSS issues and assess the needs of impacted areas, it helps to identify the impact areas and address certain questions.

Examples of Impacted Areas

The following list provides examples of areas that could potentially be impacted by the use of FOSS and questions that can be used to help assess the needs of these areas:

Project/Program Methodology	• What criteria are used to review the use of FOSS in development processes? • What methods are used to assess product quality versus deployment and development quality? • What program complexities are incurred with each incremental inclusion of additional FOSS technologies?
Human Capital	• What should an organization's policy be governing employee participation on FOSS projects? • Where does open source participation and use tie into an individual's performance plan? • When an employee participates in FOSS development, what are the guidelines separating personal versus organizational contribution?
IT Infrastructure	• How do FOSS deployment, distribution, and management processes change within the existing IT infrastructure management?
IT Procurement	• What changes to the procurement processes are needed to capitalize on FOSS? • How does procurement ensure that a supplier's products are not in violation of FOSS licenses?
IT Outsourcing	• What are the implications with regard to FOSS if the organization wants to outsource work? • How can an organization mitigate the attendant risks? • When FOSS development is outsourced, what policies need to be in place with the outsource company defining the use of FOSS? Can the outsource company use FOSS for their organization or other customers? • What licenses are acceptable in outsourcing? If FOSS is used, then what obligations does the outsourcing company have in keeping the code confidential?
Software Development Life Cycle	• What modifications are needed for the Software Development Life Cycle (SDLC) to leverage FOSS? For example, if an organization buys a product from a supplier, the organization typically monitors the viability of the vendor. • How does the organization monitor the viability of the open source community or product itself?

IT Portfolio Management	• What changes are necessary to criteria used for managing the IT portfolio, especially the technical fit, to properly compare FOSS solutions with vendor supplied or in-house development?

Understanding FOSS

For software to be defined as open source, it must meet a strict set of guidelines established by the Open Source Initiative. Licenses certified as open source must also provide a set of rights to the user as specified in the definition. The official definition of open source can be found at:

http://www.opensource.org/osd.html

Three Basic Areas of FOSS

There are typically three aspects of open source: licensing, community, and methodology. A better understanding of these aspects can enable you to properly use and benefit from FOSS in your organization. The following section lays the groundwork for understanding open source by exploring licensing, community, and methodology.

Licensing

Licenses are obligations set forth by owners of a particular work, such as software, that govern the use of their work. Many types of licenses exist today and various organizations within the open source development community have differing beliefs regarding licensing and how to use it in their work.

FOSS licensing is often overlooked by developers and IT management. As the number of licenses within an organization continue to grow, it becomes even more important to understand FOSS licenses to avoid violations that could result in litigation or loss of business. The nature of open source licensing presents unique challenges. The major challenge is knowing what the specific license requirements are and, whether you are in compliance. Understanding the context within which FOSS licenses are used must also be examined.

Unlike commercial software licensing, open source has many people contributing to the work for free, many people holding copyrights to their work, and restrictions exist for some open source software that deal specifically with how it interacts with other software. Consequently, it is important to understand what rights and restrictions exist when using a particular piece of FOSS. In addition, some open source licensing terms do not typically exist in commercial licenses, making it important to understand these terms. For a list of current, approved open source licenses by category, see the Open Source Initiative web site located at:

http://www.opensource.org/licenses

Open source software can also place license restrictions on the open source code's interaction with other software. For example, multiple open source licenses can be embedded in larger software and/or hardware products. These interactions can be complex as projects grow into multiple layers of code. It can be difficult to verify that the license provided with the software reflects all of the lower-level licenses that are embedded within it.

Some open source projects are performed by organizations who control all copyrights to the code. This allows organizations, such as MySQL, to follow a dual-licensing strategy: MySQL offers their product as open source, but companies who do not want to follow the open source license can obtain a different license form the company. In the case of the MySQL database, this is useful for organizations that want to incorporate the database into their product.

Community

Another key aspect of open source software is the community. An open source community is a collection of developers and users with a common interest in the creation, enhancement, and support of a specific piece of open source software. Historically, the community was comprised of many “free agents” or individual contributors. One of the main attractions to the open source community is the ability to customize open source software to meet an individual's needs. Once modifications are made, new features can be released back into the community. Another benefit of the open source community is the large network of peer reviewers. Because open source code must be accepted and reviewed by a large peer community, the code is generally written in a sound manner with strict adherence to coding standards. With the input and scrutiny of the open source community, the resulting software can often be more robust.

As the open source community evolves, FOSS activities are increasingly funded by large organizations which share development costs. For example, HP invests in Linux to improve the operating systems performance on their hardware. IBM invests in Linux because it provides an opportunity to use a common development set across all their platforms and for other business reasons. Government and academia are also contributing to the open source community at an increasing rate.

Participation in the open source community can bring many benefits to an organization's business model. Organizations can use open source software to improve development models, minimize development costs, and decrease time to market for software or products. Organizations have also discovered that some goals can be achieved faster using open source code as building blocks.

Methodology

The final aspect of open source is the general methodology that guides open source development by the community. There is more than one open source methodology. However, there are certain characteristics shared by all, such as open development. The specific methodology can vary depending on the project. The methodologies can be very different than organizations might expect. When organizations use FOSS, they like to know the community's plan for evolving the software. Very few road maps currently exist, but some projects are starting to create and publish them. Influence and control over the use of FOSS can be achieved by integration and involvement of the individuals who are largely in control of the development of FOSS. Different projects generally have their own subculture which has an impact on the governance methodology that is used.

An organization's involvement in open source projects through their employees must be done in alignment with the culture of each project. It is important to understand that participation and merit is earned; it stays with the individual and not the organization. When you have a high-performing employee who is working on a project, the business is heavily invested in the outcome of that project. If that employee leaves the organization, the ability to work within that community moves with that employee.

The following examples show two significantly different open source development methodologies.

- The “benevolent dictator” methodology can be seen in use for the Linux kernel. Linus Torvalds initiated the development of the Linux kernel, leads development activities, and makes key decisions. However, he appoints “lieutenants” that oversee large portions of the development. One of his key mottos is “simplicity over performance.” This autocratic development methodology has proven over time to be a successful management strategy for dealing with contributions to the Linux kernel from a variety of competing development groups.
- A well-documented governance methodology can be seen in use at the Apache Foundation. The Apache Foundation has specific documents that define their methodology for related contributions to the Apache project, overall governance rules, and in general, how the Apache project is managed.

FOSS Policies and Guidelines

A consistent set of policies and guidelines is needed to govern FOSS within an organization. These policies and guidelines must be carefully developed to insure that all issues that may affect the interests of the organization are addressed. Because an open source governance policy is a key business management process, it is recommended that all FOSS policies are developed jointly with other departments in your organization which might be impacted, such as legal, business management, IT management, and engineering.

This section provides guidance for developing an open source policy and describes the rationale behind various aspects of the policy, such as motivations, business justifications, and open source community participation. Additional information that can be used when considering FOSS policies are discussed. These components of an open source policy include: business use cases, appropriate and inappropriate uses of FOSS and associated projects, employee participation, and other considerations.

General Guidelines for FOSS Policies

A few general guidelines that should be considered and addressed when developing a FOSS policy include:

- Determine if the use of FOSS is beneficial. This can be examined by defining the business need, the role it plays in infrastructure and development projects, and cost saving trade-offs.
- Develop a process for FOSS governance. This includes policies, procedures, and tools to manage the acquisition, use, licensing, deployment, and distribution of FOSS. Within the context of this process, employee and management responsibilities should be defined, specifying their roles and responsibilities in support of the FOSS governance strategy.
- Define the extent to which an employee can contribute to open source. Provide specifics for how, when, and where an employee should bring FOSS and its related projects into the organization and how much they should participate in the related community.
- Determine relationships with the open source community. Consideration must be given to how your organization works with the open source community, in general and in relation to specific FOSS projects. Develop best practices for managing the relationship. This should include organizational policies for licensing FOSS projects and protecting proprietary assets.
- Develop a documentation plan in support of communication and awareness of the organization's FOSS governance strategy. In addition to traditional documentation, this may include training, internal public relations campaigns, and other educational opportunities.

Components of FOSS Policies

There are many things FOSS policies should address, and questions that need to be answered. The policies needs to go beyond just understanding the license requirements and answer some of the following questions:

- How is FOSS chosen?
- How is FOSS acquired?
- How and where is FOSS used?
- How is FOSS supported?
- How is FOSS tracked and how are FOSS projects tracked?

Choosing FOSS

It is crucial to perform an analysis of FOSS candidates to insure potential risks are clearly understood and documented. This analysis includes a determination on whether the open source software is designed for a specific use case; and if so, whether the licensing terms reflect this purpose. You should determine if the use of FOSS for a particular project is “mission critical.”

To determine if a particular piece of FOSS is a good fit for your organization, information should be obtained and evaluated regarding the particular open source project community. This requires delving into open source community issues such as the size, age, and activity level within the community. The “health” of the community and its goals are also important.

Another important factor to examine is the legal status of a FOSS project. You should look for both current litigation and the potential for future litigation. There are many resources available on the web that can assist in your evaluation of FOSS. For a list and description of some of these resources, see the FOSSBazaar web site located at:

http://www.fossbazaar.org/?q=resources

Acquiring FOSS

Once it has been determined that a particular piece of FOSS is to be used within your organization, the next step is defining the process for acquiring the software. Knowing where to retrieve the software is only the first step. In addition, you should verify the following:

- The open source code came from a single, authentic source repository.
- The open source bits arrived intact.
- The open source is not corrupt.
- The open source code was compiled from the source; if not, verify there is no malware embedded in the binaries.

Using FOSS

It is important to know how open source will be used in your organization in order to understand both the short and long-term ramifications. This section discusses the types of questions that you need to ask to clearly understand how FOSS will be used.

How FOSS will be used can come into play when addressing a variety of FOSS licensing issues. It should be determined if the open source software is currently used with other proprietary code and whether it will be used this way in the future. You should determine if the license allows for the use for which you intend. Not all FOSS licenses are compatible with one another. As licenses continue to proliferate and become more complex, more and more pieces of software are becoming incompatible so you cannot always combine two different pieces of software for different purposes.

Next, you should find out how many groups use each FOSS component. It is important to determine if there are other groups in the organization using similar but different components, or different versions of the same component. The open source policy needs to provide a mechanism, such as a database, that can be used to inventory, track, and manage all FOSS use within the organization. A common complaint from production teams is that development teams are using different open source software to perform the same fundamental functionality, or they are using different versions of the software. For example, Apache Geronimo and JBoss JEE Application servers have very similar functionality. The organization must put controls in place for different versions of software that are in use to avoid unnecessary work maintaining multiple versions.

The use of FOSS can be valuable to the operations of an organization for reasons other than specifically generating revenue. For example, turning software over to the open source community can be used as an exit strategy to enable the user community to continue to enhance and maintain the software.

Another benefit of using FOSS is the ability to distribute development costs across multiple organizations with no loss of competitive advantage. When an organization wants to be an active participant in a FOSS project, it is more cost-effective to build non-differentiating technology so the organization can spend more of their resources building innovative technologies that provide differentiation. There are numerous new efforts aimed at developing open source consortiums so vertical markets can leverage each other for code and technology but without any differential advantage.

Supporting FOSS

Determining how to support FOSS use in an organization is critical. Every employee within the organization should adhere to the established guidelines. There are three different approaches that are commonly used when supporting FOSS:

Self Support	The decision as to whether FOSS can be self-supported is dependent on the quality of the ties to the specific open source community; employee participation in the open source community; and ultimately, whether a specific plan can be developed for self support.
Commercial Support	Commercial support can be provided to an organization through vendors such as HP, Red Hat, Novell, MySQL, Symas, and so on. This type of support is used when an organization has a direct, vested interest in the project. The organization typically employs contributors or people who can focus on the maintenance of specific components. The relationship between the different vendors is important. When considering the use of commercial support, you should determine if a positive working relationship exists with the vendor and whether some level of integration support exists. The stability of the organization providing support should also be considered.
Commercial Support by Integration Vendors	An organization can use integration vendors to provide commercial support such as HP, OpenLogic, SourceLabs, Spike Source, Covalent, and so on. Integration vendors may not be major contributors or to the development or maintenance of the open source, but they still have some level of investment in those projects. They also play a role in a larger set of projects and perform integration testing.

Tracking FOSS

Part of an ongoing maintenance policy for using FOSS in your organization is determining how the FOSS project is to be tracked over time. Unlike commercial software vendors with one or only a few vendors to track, FOSS has numerous entities that must be monitored. This reflects a different tracking process than what has traditionally been used by organizations.

It must be decided who is going to track certain aspects of the open source. Important issues to track include vulnerabilities and critical defects that might be posted against a particular piece of FOSS. If possible, these defects should be integrated into the organizations's development cycle. The health and road map of the FOSS project should also be tracked. Potential problems to look for are off shoots from the original project, discontent within the community, and whether major contributors are leaving. How often the health of a FOSS project should be monitored depends on the particular project.

In theory, internal FOSS should be controlled by restricting access to open source repositories. However, in practice, successful implementation of such a policy has not yet worked. With a prolonged process for the submission and approval of a FOSS request, it is inevitable that the members of the organization will circumvent the process and obtain FOSS in a more timely fashion.

Determining Potential Business Use Cases for FOSS

FOSS can be advantageous for an organization and can have cost-savings benefits. Additionally, there are a number of other business use cases that might not be as obvious. The following is a list of use cases to consider when determining whether your organization should contribute to the FOSS community.

- Establishing your FOSS implementation as an industry standard. An organization can maintain a controlling interest in the FOSS project while still allowing others to view, modify, and use the particular piece of FOSS.
- Increasing sales of other products including hardware and software. Vendors such as HP leverage hardware sales by modifying Linux and other FOSS projects so they are compatible

with their hardware. This can also lead to wider acceptance of the open source in the marketplace.

- Distributing the expense of FOSS maintenance among other collaborators. Because the use of FOSS is so pervasive in product development today, this distribution of costs to a larger community can offer significant cost savings as well as leading to the development of a more robust product.
- Gaining cooperation from the open source community. Some organizations may become involved by taking the lead in a FOSS project while others may choose to leverage the results that are gained through open source community cooperation.
- Providing a strategy for a product's end-of-life plan. This is can be an attractive alternative for an organization. By open sourcing software that was previously proprietary, product support can be continued through the open source community. Users are given access and self-support capabilities.
- Enhancing an organization's image in the marketplace. Advertising your organization's involvement in open source can improve your image, resulting in a more "high-tech" image. By leading or participating in the development of FOSS, you can generate favorable publicity among peer organizations and the open source community. This can also attract new hires.

Policies and Guidelines for Employee Open Source Contributions

Because there are many ways in which an employee can make contributions to open source, thought should be given to developing guidelines and policies related to these contributions. This section discusses potential issues to consider when developing these guidelines and policies.

First and foremost, an organization should provide documentation and training on their FOSS policy. It is only with a clear understanding of the policy that employees can be held responsible for compliance.

Next, a procedure should be implemented through which an employee can submit a proposal for management approval for using or developing FOSS.

Employees should understand they are responsible for disclosing any involvement with FOSS projects, regardless of whether they consider it work for the organization or personal work outside of their employment. Many large organizations have clauses in their employment contracts, particularly for software development, that anything the employee develops is an asset of the organization. In some cases, even though an employee may perform work on FOSS projects outside the realm of their employment, they may still be obligated to the organization's policies and licensing requirements. In addition, it is not always clear who holds the copyrights to FOSS contributed by an employee. Copyrights may belong to the employee, the organization, or both. An employee is also responsible for protecting an organization's proprietary information and trade secrets.

Work performed on a FOSS project has the potential for conflicts of interest. One way in which a conflict can exist is how closely the external work of the employee is tied to their job within the organization. An overlap between the external open source work of an employee should, in most cases, not overlap with their internal company projects. For example, if the organization is working to create a product to compete with one that already exists in the open source community, it could be a problem if an employee is also working towards the same goal on personal time. Another conflict of interest could occur when an employee works on a particular FOSS project while employed with the company, but has the intent of leaving the organization to build a business that supports the open source software he has worked on.

Conversely, there may be reasons why an organization would encourage employees to work on their own time on a FOSS project. For example, it would be beneficial for an employee to work externally on a FOSS project which could compete with another vendor's proprietary offering. Additionally, when an organization introduces legacy software into the open source community when it is at the end of its life cycle, an employee with intimate knowledge of the software could make external contributions as a member of the open source community at large.

Another issue regarding an employee's contributions to FOSS projects, is the consideration of what is appropriate use of an organization's time versus employee personal time. This includes organizational resources such as e-mail, internet access, phone, computer equipment, and so on.

Other Open Source Policy Considerations

There are some less obvious, but equally important components that should be taken into account when developing your organization's FOSS policy. This section discusses other guideline considerations.

Relationship with the Open Source Community

Maintenance of positive working relationships within the open source community is very important. To develop appropriate guidelines, you should determine the objectives of the organization's relationship within the open source community. Organizations often lose credibility when they create business models around generating revenue through a FOSS project in such a way that they are not sufficiently giving back to the community. Organizations should retain software that differentiates themselves from their competitors. Part of the policy should be to distinguish between software that provides a competitive advantage and software that is "common" and could be beneficial to many. The result is less software that must be maintained solely by your organization. If code is maintained locally, it substantially increases the level of effort required to merge the code into new releases of the FOSS.

Software developers in the organization also have certain roles and responsibilities in their interactions with the open source community. Policy considerations should include how to work on disclosed FOSS projects, performing work in the community where a third party might have intellectual property rights within a collaborative effort, and tracking their ability to access and leverage of FOSS.

Guidelines should be developed regarding IT's roles and responsibilities for acquiring, testing, and using FOSS and interacting within the open source community. It is important to respect other people’s time by performing due diligence on your work before it is presented. This is one way in which you can gain credibility, respect, and have a stronger voice within the open source community.

Competing with existing FOSS projects is typically viewed as inappropriate in the open source community. However, there are certain instances when competing with an existing FOSS project can be beneficial. Ultimately, there needs to be a compelling reason for creating competition. For example, competition can be appropriate if the direction of the existing project is not the same as the organization's project, or if the party responsible for maintaining the existing project is unwilling to accept your contributions.

Copyrights and Patents

Without knowledge of who owns the copyrights and patents for a particular piece of FOSS, it is difficult to understand when copyright or patent infringement occurs. You must recognize that software patents are a concern for any software consumer, independent of whether the software is open source or commercial. Commercial entities offer some level of indemnification, although they typically have a limit on how much they are willing to pay. No company can fully indemnify anybody else. For example, consider an organization using a commercial software package in a mission-critical environment. If that software is found to infringe on patents to the extent that the organization can no longer use the software, the potential impact on business is enormous. Recognize the risk and exposure presented by using FOSS and mitigate to the extent possible.

Even with a policy in place, you are still at risk for committing patent infringement. In the open source community there are a variety of ways you can mitigate patent infringement risk such as using an IP company such as the Open Invention Network (OIN) which is stockpiling Linux-related patents for the purpose of minimizing IP issues related to using Linux.

FOSS Management

This section focuses on FOSS management issues and recommendations for effective management strategies, such as developing an FOSS program office, creating an open source review board, maintaining a FOSS database, and tracking compliance with FOSS licenses.

The specific steps required to manage FOSS in an organization differ from management strategies used for traditional software governance. FOSS and associated licensing require a management program that is designed to handle the complexities inherent in the use of open source. The type of issues that a management plan needs to address are, first and foremost directed by the nature of the organization. Management strategies for an organization that focuses on software development can differ from strategies for an organization that primarily uses FOSS. In most IT organizations, the management strategy needs to cover both use cases.

Their are many reasons to effectively use and manage FOSS:

- To boost engineering productivity
- To lower an organization's total cost of operation
- To increase software choices and move towards vendor neutrality
- To manage legal risks
- To maintain competitiveness
- To maintain good standing among the open source community by respecting the spirit of open source

FOSS Management Issues

The uncontrolled proliferation of FOSS within an organization is an enormous issue with far-reaching consequences, yet it is largely unaddressed by organizations today. Because FOSS can enter your company without a formal purchase agreement, the traditional procurement process is normally bypassed-skipping finance, contract, legal, and negotiation. Even though FOSS code is available without cost, it still comes with a license agreement that imposes contractual obligations on those who use it and/or distribute it. Without careful management and adherence to open source license obligations, organizations can inadvertently expose themselves to potential litigation.

The following is a list of key open source management strategies that can be used to address and mitigate the potential risks of using open source:

- Create a set of policies that regulate how an organization deals with FOSS.
- Establish a clear set of procedures that an employee must adhere to in order to acquire FOSS.
- Establish a formal FOSS governance process with governing authorities such as an open source review board (OSRB).
- Compile and track inventory of all FOSS and related projects including: deployment, code use and reuse, community contributions, and shipments of products containing FOSS.
- Establish an internal open source community to provide organizational guidance and leadership and to manage utilization and propagation of FOSS technology.
- Develop FOSS legal expertise within your organization.
- Define and communicate to all employees the organization's FOSS policies and guidelines through cross-organizational training and awareness programs.
- Establish open source solutions that are approved by management.
- Leverage the experience of other organizations, such as HP, that have implemented an effective FOSS governance strategy. The FOSSBazaar web site is a good place to find this type of collaboration information which also leverages member expertise.

Open Source Program Office

An open source program office acts as the focal point for the overall management of FOSS governance issues. The purpose of the open source program office is to manage FOSS at every point it touches in the organization to optimize value and reduce risk. To address these issues, an open source program office is responsible for developing and managing the following: open source policy documentation, FOSS review processes, a FOSS database, compliance tools, and an organizational training program.

The main function of the open source program office is to direct and oversee all FOSS activities. It is responsible for working to define what appropriate FOSS usage is for the organization and ensuring that policies and guidelines are developed and implemented towards that end. The open source program office should also provide consistent internal and external communication regarding the organization's FOSS policy.

After the open source program office is created, one of the very first steps in its governance strategy is to oversee a comprehensive audit of the use of FOSS in the organization. The audit should identify and capture information on all FOSS in use throughout the organization, whether it is for internal IT infrastructure, redistribution to customers, part of an open source project, or embedded in other products. It is also important to work with partners and suppliers to determine the use of FOSS in third-party products that are being integrated or distributed with products from your organization.

FOSS Policy Document

Based on the information gained from the FOSS audit and combined with internal organizational objectives, a FOSS policy document should be developed. Development of the policy document should be directed by the open source program office in a joint, cross-organizational effort that includes legal, business management, and IT strategy. Important questions that need to be addressed include:

- How to determine appropriate use for FOSS?
- How should employee contributions be handled?
- What governance issues require tracking?
- What tracking processes should be implemented?
- What is the organization's relationship with the open source community?
- What are the different use cases for a particular piece of FOSS?
- How should the organization work with third parties or competitors on a community project?
- How should the organization address license and legal challenges?

The FOSS policy manual provides guidelines for working within the open source community, as well as a detailed list that can be used to evaluate the appropriateness of bringing a particular piece of FOSS into the organization. Once a policy document is in place, an organization can more effectively build an environment for developers consisting of a rich ecosystem of tools and software components; some of which can be created using existing work from the open source community. Guidelines in the policy document should also help to protect an organization from a variety of potential pitfalls that are associated with using FOSS.

Once a policy has been defined and communicated, the next step is the analysis of the different licenses for all FOSS. The license analysis is performed by the organization's legal department. Subsequently, an Open Source Review Board (OSRB) can be established to assist to determine licensing requirements and verify that FOSS components are being used in compliance with their associated license requirements. The following section explores the role of an Open Source Review Board.

Open Source Review Board

One of the main functions of an Open Source Review Board (OSRB) is enforcing the organization's FOSS policies. An OSRB is a virtual team of open source experts, including IT, legal, engineering, and community members. It is also the job of the OSRB to determine if due diligence was performed for the FOSS proposals. The OSRB provides a centralized resource to answer questions and monitor FOSS activity within an organization. Specific responsibilities of the board include reviewing and monitoring some of the following:

- Appropriate use of FOSS in an organization's products, solutions, and services
- Group and individual participation in FOSS projects
- Use of an organization's software when released under an open source license
- Internal use of FOSS for IT infrastructure
- Adherence to the established open source proposal workflow

The ultimate goal of the OSRB is to facilitate FOSS use and employee participation while mitigating risks. Given the complexity surrounding FOSS licensing and the potential interaction between open source and proprietary code, it is not surprising that FOSS proposals require intensive analysis. To make the license analysis process more efficient, tools are available that automate license discovery and pre-classification. To streamline the process, the OSRB can develop a process for proposal development and review. The OSRB can also use a database and other open source tools to automate the process.

FOSS Proposal and Review Process

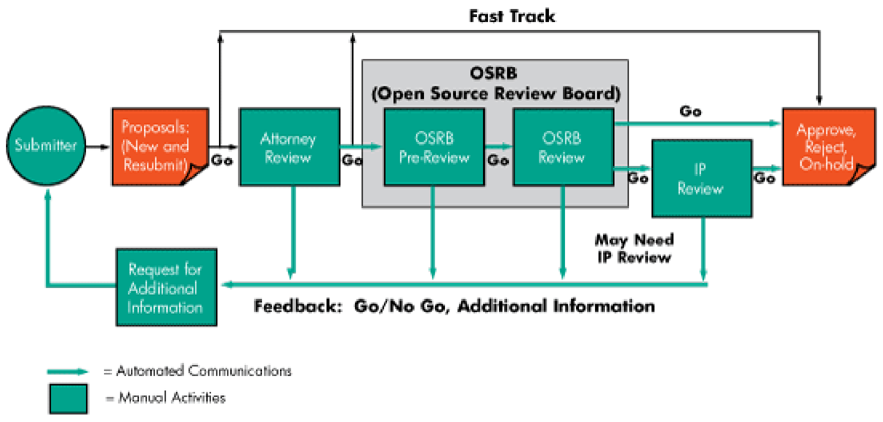
When an individual employee or a project team wants to get approval from the OSRB to use FOSS, a proposal must be submitted. A proposal template can be used to help insure that all pertinent information is provided to the OSRB. Basic information is necessary, such as proposal owner, management approval, and an overview of the FOSS proposal including an architectural diagram and/or design documentation. The following is a partial list of topics that should be addressed in the proposal:

- Describe how the open source will be used and the business use case.
- Examine the licensing requirements and patent guidelines.
- Quantify the total cost of ownership (TCO) and return on investment (ROI) .
- Define employee, management, and community responsibilities.
- Describe documentation that will be developed to increase organizational awareness
- Describe how the software will be supported.
- Describe how the software will be managed and tracked over time.
- Determine if the open source be used as an advertising tool
- Provide a list of all FOSS in the proposed product.
- Describe the methods that will be used to create and deliver the FOSS project.

OSRB Proposal Workflow

A logical workflow should be developed to outline the process through which an individual employee or project team can get approval to deploy or develop FOSS. Figure 1 details a high-level, generic workflow diagram. The process flow for your organization will include many steps that are specific to your business. The OSRB proposal workflow contains a set of information gathering steps that can assist in the creation of a clear and concise picture of the impact the open source would have on your organization and the potential risks.

Figure 1 Generic Open Source Review Board Workflow


Fast Track
OSRB
(Open Source Review Board)
Submitter
Proposals: (New and Resubmit)
Go
Attorney Review
Go
OSRB Pre-Review
Go
OSRB Review
Go
Go
IP Review
Go
Approve, Reject, On-hold
May Need IP Review
Request for Additional Information
Feedback: Go/No Go, Additional Information
= Automated Communications
= Manual Activities

The sequence of tasks in the OSRB proposal workflow are designed to verify that all the required information has been provided and to perform specific evaluations of the proposal. At each step, there is a feedback loop that can be used to obtain more information from the submitter. The essential steps in the proposal workflow are as follows:

1. Prepare a proposal that includes information regarding the specific piece of FOSS. Three basic questions should be answered: What is the software that is being submitted for evaluation? What license agreement is associated with this particular piece of FOSS? and, What is the intended use for the open source?

   This proposal can take many forms but it is recommended that a template be created and that it is required for a submittal. This can help to insure that all crucial information for the FOSS evaluation is included.

2. Conduct a preliminary legal review by an attorney to insure that all necessary legal and licensing information is included. This preliminary review screens proposals early in the process so proposals with inadequate or missing information are sent back for rework before they reach OSRB.
3. The review is performed by the OSRB. If the OSRB's review of the proposal indicates the need to perform an Intellectual Property (IP) review, the proposal is then forwarded to an IP attorney for review. An IP review is used to check whether a proposed project can be done as open source or whether there are other opportunities to commercialize the IP. If an IP review is not deemed necessary, the OSRB can make the final determination on whether the proposal is approved.

The workflow also includes a “fast track” as some proposals are more straightforward than others or are very similar to previous proposals. For example, an MIT license is very permissive, because of this you may choose to quickly approve proposals that are only using software under that license. It can be valuable to document all FOSS projects used within your organization, but the level of scrutiny and understanding of each depends on the licenses contained within the proposal. Specifically, if a team wanted to embed the Apache web server in a product, many of the OSRB steps can be skipped, allowing for a quicker review by the OSRB.

Open Source Review Board Database

Because it provides an inventory of FOSS projects in use throughout an organization, the OSRB database is a vital component of the FOSS governance process. In addition to tracking FOSS, the OSRB database can assist in identifying specific FOSS that could be used to handle issues such as security, license changes, and potential patent issues. This information can then be communicated to the appropriate individuals or teams within the organization.

Open Source Compliance Tools and Portals

For the OSRB to be successful, tools are needed when possible to automate a process. The OSRB often has to examine all source files and packages, know what they are, and document them; this can be a tedious and difficult task. Without automated tools, it would be difficult, if not impossible, to track the use of FOSS in an organization and to address the wide range of open source issues.

There are a small number of automated tools currently in existence. Every organization should evaluate for themselves which open source compliance tools are available and which represent the best available technology for their business. Many organizations, such as HP, currently use tools that are developed internally. The open source FOSSology project is a good example. HP decided to open source its FOSSology tool, which is based on internal tools that HP developed for license identification. For more information on FOSSology, see the web site located at:

http://www.fossology.org

While most developers have a general understanding in regard to FOSS licenses and legal issues, an OSRB attorney or other legal expert is ultimately needed. These experts can spend anywhere from several weeks to several months on complex proposals. This is another reason why an automated tool is necessary to expedite the process. There are two main functions that any open source compliance tool should perform: License discovery and analysis, and code reuse detection of both source and binaries. The following sections explain these functions in more detail.

License Discovery and Analysis

Depending on the size of the organization and the number of FOSS products in use, the number of licenses that require tracking can be in the hundreds. Even within individual software packages, there can be multiple licenses. These embedded licenses can be especially tricky to find and track. With an automated compliance tool, you can perform some of the following searches:

- Find common license terms and phrases
- Find known licenses
- Find new licenses

By identifying licenses quickly, it can reduce the chance of compliance issues and facilitate a quicker turnaround on FOSS proposals. Automating license discovery and analysis benefits both the submitters and the reviewers of FOSS proposals. In addition to the large number of FOSS licenses, it is very common for one license to exist with two different names; both having identical text. Additionally, independent projects can change the wording of a license and add or remove existing conditions, creating subtle differences.

Online resources exist which track a large number of licenses that are in effect today. These should be used as complementary resources along with automated compliance tools. One valuable resource that can be used in the process of license discovery and analysis is the Open Source Initiative (OSI) web site. The OSI maintains a list of both approved open source licenses and licenses that are still in the approval process.

You should exercise caution with any open source compliance resource, automated or online. When using these license analysis and discovery resources, some consideration should be given to the following:

License Claims Should Not Automatically Be Trusted	You can never rely solely on the open source to provide accurate licensing information. Every organization must research and verify these for themselves. Independent verification of FOSS licenses should include determinations such as: • Is the license for the entire open source package? • Has the license been previously verified? By whom?

	• Does this license include other components that are under different licenses? • Is there any interaction between the FOSS code and proprietary code?
Licenses Evolve and Change	Even though a license has been encountered once, it should still be researched to verify that no changes have been made since that time. Additionally, organizations should note that they can work within the open source community to change a license.
License Analysis Requires Both Legal and Technical Knowledge	The license review should be performed by a multidisciplinary team, such as the OSRB, with includes both legal and technical expertise.
License Preferences and Policies Can be Influenced by Key Business Drivers and Practices	When analyzing a license for use, there are some questions to consider: • Is the FOSS development part of an outsourcing agreement or partnership? • Are patents important? • Do the developers understand the nuances of the licenses?
A Newer License Version is Not Always Better	Do not assume that just because an updated version of a license exists that it is "new and improved". For example, the GNU Lesser General Public License (LGPL) version 2.1 is compatible with both GPLv2 and GPLv3. However, the newer version, LGPLv3 is only compatible with GPLv3 and not GPLv2 by itself and this newer version is generally recommend for special circumstances only.
Create a Backup for License Discovery and Analysis	As with other business critical data, a backup should be created for the license discovery and analysis information. The license data should also be continuously updated with changes to the licenses and newly discovered licenses. You can leverage previously analyzed licenses when performing new analyses and focus on identifying changes that may have occurred between versions of FOSS.

Source Code Reuse Detection

It is common practice in software development to actively encourage the reuse of source code. This increases productivity and prevents employees from reinventing the wheel. Source code is more accessible for reuse since it can be found using popular search engines, making high quality code more readily available for various tasks. The following are potential scenarios in which source code reuse may be of value:

- Redistribution of products that are spin off units.
- Acquisition of software product. As a part of mergers and acquisitions activities and OEM deals.
- Developing software products with partners or in outsourced development projects.

Conclusion

The use of FOSS is pervasive and inevitable in most large organizations. A proactive approach to FOSS management is critical; not only in preventing legal and loss of business issues, but in maximizing the benefits of using FOSS in your organization. With an understanding of FOSS governance fundamentals, you are ready to move forward and begin creating an action plan aimed at the development of a comprehensive FOSS governance strategy.

For additional FOSS Governance information, see the FOSSBazaar web site located at:

http://www.fossbazaar.org