OPENCHAIN

CURRICULUM

Core FOSS Compliance Version 1
Designed for Version 1 of the OpenChain Specification

This is not legal advice

Contents

1. What is Intellectual Property?
2. Introduction to FOSS Licenses
3. Introduction to FOSS Compliance
4. Key Software Concepts for FOSS Review
5. Running a FOSS Review
6. End to End Compliance Management (Example Process)
7. Avoiding Compliance Pitfalls

CHAPTER 1

What is Intellectual Property?

What is “Intellectual Property”?

- Copyright: protects original works of authorship
  - Protects expression (not the underlying idea)
  - Software, books, audiovisual materials, semiconductor masks
- Patents: useful inventions that are novel, useful, non-obvious
  - Limited monopoly to incentivize innovation
- Trade secrets: protects confidential and valuable information
- Trademarks: protects marks (word, logos, slogans, color, etc.) that identify the source of the product
  - Consumer and brand protection; avoid consumer confusion and brand dilution

This chapter will focus on copyright and patents, the areas most relevant to FOSS compliance

Copyright concepts in software

- Basic rule = copyright protects creative works
- Copyright generally applies to literary works, such as books, movies, pictures, music, maps
- Software is protected by copyright, not the functionality (that’s protected by patents) but the expression (creativity in implementation details)
- The copyright owner only has control over the work that he or she created, not someone else’s independent creation

Copyright rights most relevant to software

- The right to *reproduce* the software – making copies
- The right to create "*derivative works*" – making modifications
  - The term derivative work refers to a new work based upon an original work to which enough original creative work has been added so that the new work represents an original work of authorship rather than a copy (note that this is a term of art under US law)
- The right to *distribute*
  - Distribution is generally viewed as the provision of a copy of a piece of software in binary or source code form to another entity (an individual or organization outside your company or organization)

Note: The interpretation of what constitutes a “derivative work” or a “distribution” is subject to debate in the FOSS community and within FOSS legal circles

Patent concepts in software

- Patents protect functionality - this can include a method of operation, such as a computer program
  - Does not protect abstract ideas, laws of nature
- The patent owner has the right to stop anybody from exercising that functionality, regardless of independent creation
- Other parties who want to use the technology may seek a patent license (which may grant rights to use, make, have made, sell, offer for sale, and import the technology)

Licenses

- A "license" is the way a copyright or patent holder gives permission or rights to someone else
- The license can be limited to:
  - Types of use allowed (distribution, derivative works / to make, have made, manufacture)
  - Exclusive or non-exclusive terms
  - Geographical scope
  - Perpetual or time limited duration
- The license can have conditions on the grants, meaning you only get the license if you comply with certain obligations
  - E.g, provide attribution, give a reciprocal license
- May also include contractual terms regarding warranties, indemnification, support, upgrade, maintenance

Check Your Understanding

- What type of material does copyright law protect?
- What copyright rights are most important for software?
- Can software be subject to a patent?
- Does a patent give rights to the patent owner?
- If you independently develop your own software, is it possible that you might need a copyright license from a third party for that software? A patent license?

CHAPTER 2

Introduction to FOSS Licenses

FOSS Licenses

- Free and Open Source Software (FOSS) licenses generally make source code available under terms that allow for modification and redistribution
- FOSS licenses may have conditions related to providing attributions, copyright statement preservation, or a written offer to make the source code available
- One popular set of FOSS licenses are those approved by the Open Source Initiative (OSI) based on their Open Source Definition (OSD). A complete list of OSI-approved FOSS licenses is available at http://www.opensource.org/licenses/

Permissive FOSS Licenses

- Permissive FOSS license - a term used often to describe minimally restrictive FOSS licenses
- Example: BSD-3-Clause
  - The BSD license is an example of a permissive license that allows unlimited redistribution for any purpose as long as its copyright notices and the license's disclaimers of warranty are maintained
  - The license contains a clause restricting use of the names of contributors for endorsement of a derived work without specific permission
- Other examples: MIT, Apache-2.0

License Reciprocity & Copyleft Licenses

- Some licenses require the distribution of derivative works (or software in the same file, same program or other boundary) under the same terms as the original work
  - This is referred to as a "copyleft", "reciprocal", or "hereditary" effect
- Example of license reciprocity from the GPL-2.0:

"You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed...under the terms of this License."

- Examples: all versions of GPL, LGPL, AGPL, MPL, CDDL

Proprietary License

- A proprietary software license (or commercial license or EULA) has restrictions on the usage, modification or distribution of the software
- Proprietary licenses often involve payment or a license fee
- Proprietary licenses are unique to each vendor - there are as many variations of proprietary licenses as there are vendors and each must be evaluated individually
- FOSS developers often use the term "proprietary license" to describe a commercial non-FOSS license

Other Licensing Situations

- Freeware - software distributed under a proprietary license at no or very low cost
  - The source code may or may not be available, and creation of derivative works is usually restricted
  - Freeware software is usually fully functional (no locked features) and available for unlimited use (no locking on days of usage)
  - Freeware software licenses usually impose restrictions in relation to copying, distributing, and making derivative works of the software, as well as restrictions on the type of usage (personal, commercial, academic, etc.)
- Shareware - proprietary software provided to users on a trial basis, for a limited time, free of charge and with limited functionalities or features
  - The goal of shareware is to give potential buyers the opportunity to use the program and judge its usefulness before purchasing a license for the full version of the software
  - Most companies are very leery of Shareware, because Shareware vendors often approach companies for large license payments after the software has freely propagated within their organizations.
- Freeware and Shareware are not FOSS

Public Domain

- The term public domain refers to intellectual property not protected by law and therefore usable by the public without requiring a license
- Developers may include a *public domain declaration* with their software
  - E. g., "All of the code and documentation in this software has been dedicated to the public domain by the authors."
  - The public domain declaration is not the same as a FOSS license
- Often the public domain declaration is accompanied by other terms, such as warranty disclaimers. In such cases, the software may be viewed as being under a license rather than being in the public domain

License Compatibility

- License compatibility is the process of ensuring that license terms do not conflict. If one license requires you to do something and another prohibits doing that, the licenses conflict and are not compatible
- The Free Software Foundation provides the following example to illustrate a case of license compatibility:

  A license p is* compatible with *a license q (or is* q-compatible) *if

  A work licensed under p can be distributed under the terms of q.

- Example: GPL compatibility
  - Many of the FOSS licenses, such as the MIT license and the LGPL, are GPL-compatible, meaning that their source code can be combined with source code that is licensed under the GPL without conflict; the new program resulting from the combination would have to be licensed under the GPL
  - Other FOSS and proprietary software licenses are not GPL-compatible since they have conflicting terms and conditions
  - Reference: http://www.fsf.org/licensing/licenses/

Notices

Notices, such as text in comments in file headers, often provide authorship and licensing information. FOSS licenses may also require the placement of notices in source code or documentation to give credit to the author (an attribution) or to make it clear the software includes modifications.

- **Copyright notice** - an identifier placed on copies of the work to inform the world of copyright ownership. Example: Copyright © A. Person (2016).
- **License notice** - a notice that acknowledges the license terms and conditions of the FOSS included in the product.
- **Attribution notice** - a notice included in the product release that acknowledges the identity of the original authors of the FOSS included in the product.
- **Modification notice** – a notice that you have made modifications to the source code of a file, such as adding your copyright notice to the top of the file.

Multi-Licensing

- Multi-licensing refers to the practice of distributing software under two or more different sets of terms and conditions
  - E.g., when software is “dual licensed,” recipients can choose to use or distribute the software under a choice of two licenses
- Note: This should not be confused for situations in which a licensor imposes more than one license, and you must comply with all of them

Check Your Understanding

- What is a FOSS license?
- What are typical obligations of a permissive FOSS license?
- Name some permissive FOSS licenses.
- What does license reciprocity mean?
- Name some copyleft-style licenses.
- Are Freeware and Shareware software considered FOSS?
- What is a multi-license?

CHAPTER 3

Introduction to FOSS Compliance

FOSS Compliance Goals

- **Know your obligations (detect and track use of FOSS).** You should have a process for identifying, tracking and archiving a list of all FOSS components (and their respective identified licenses) from which your software is comprised.

- **Satisfy all the license obligations for the FOSS that is used.** Your program should identify and handle typical FOSS use cases that result from your organization's business practices.

What Compliance Obligations Must Be Satisfied?

Depending on the license(s) involved, obligations could consist of:

- **Attribution and Notices.** Inclusion of copyright and license text in the source code and/or product documentation or user interface, so that downstream users know the origin of the software and their rights under the licenses
- **Source code availability.** Providing source code for original work, for combined work or modifications, as well as build scripts (scripts that control the build process)

These obligations may trigger upon key events, such as:

- External distribution
- Whether you have made modifications

FOSS Conditions & Restrictions

Depending on the FOSS license used, you may need to comply with one or more of the following types of conditions and restrictions:

- Retain copyright (and other) notices
- Provide a copy of the license
- Provide notice of modifications
- Modified versions must have a different name to avoid confusion
- Provide access to source code (whether you modified it or not)
- Maintain modified versions (derivative works) under the same license
- Provide attribution
- Do not use the project or copyright holder name or trademark
- Do not restrict others of the rights granted under the original license
- Termination clauses (if you breach, you lose license)

FOSS Compliance Triggers: Distribution

- Dissemination of material to an outside entity
  - Applications downloaded to a user's machine or mobile device
  - Javascript, web client, or other code that is downloaded to the user's machine
- For some FOSS licenses, access via a computer network can be a "distribution"
  - Some licenses define distribution to include permitting access to software running on a server (e.g., all versions of the Affero GPL)

FOSS Compliance Triggers: Modification

- Changes to the existing program (e.g., additions, deletions of code in a file, combining components together)
- Modifications may constitute a derivative work, and FOSS authors may limit or place obligations on modifications
- Modifications may trigger FOSS obligations, such as:
  - Notice of modification
  - Providing accompanying source code

FOSS Compliance Program

Organizations who have been successful at FOSS compliance have created their own *FOSS Compliance Programs* (consisting of policies, processes, training and tools) to:

1. Facilitate effective usage of FOSS in commercial products
2. Respect FOSS developer rights and comply with license obligations
3. Contribute and participate in open communities

Implementing Compliance Practices

Prepare business processes and sufficient staff to handle:

- Identification of the origin and license of FOSS software
- Tracking FOSS software within the development process
- Performing FOSS review and identifying license obligations
- Fulfillment of license obligations when product ships
- Oversight for FOSS Compliance Program, creation of policy, and compliance decisions
- Training

Compliance Benefits

Benefits of a robust FOSS Compliance program include:

- Increased understanding of the benefits of FOSS and how it impacts your organization
- Increased understanding of the costs and risks associated with using FOSS
- Better relations with the FOSS community and FOSS organizations
- Increased knowledge of available FOSS solutions

Check Your Understanding

- What does FOSS compliance mean?
- What are two main goals of a FOSS Compliance Program?
- List and describe important business practices of a FOSS Compliance Program.
- What are some benefits of a FOSS Compliance Program?

CHAPTER 4

Key Software Concepts for FOSS Review

What information do you need to gather?

When analyzing FOSS usage, collect information about the identity of the FOSS component, its origin, and how the FOSS component will be used. This may include:

- Package name
- Version
- Original download URL
- License and License URL
- Description
- Description of modifications
- List of dependencies
- Intended use in your product
- First product release that will include the package
- Availability of source code
- Where the source code will be maintained
- Whether the package had previously been approved for use in another context
- Inclusion of technology subject to export control
- *If from an external vendor:*
  - Development team's point of contact
  - Copyright notices, attribution, source code for vendor modifications if needed to satisfy license obligations

How do you want to use to the component?

Common scenarios include:

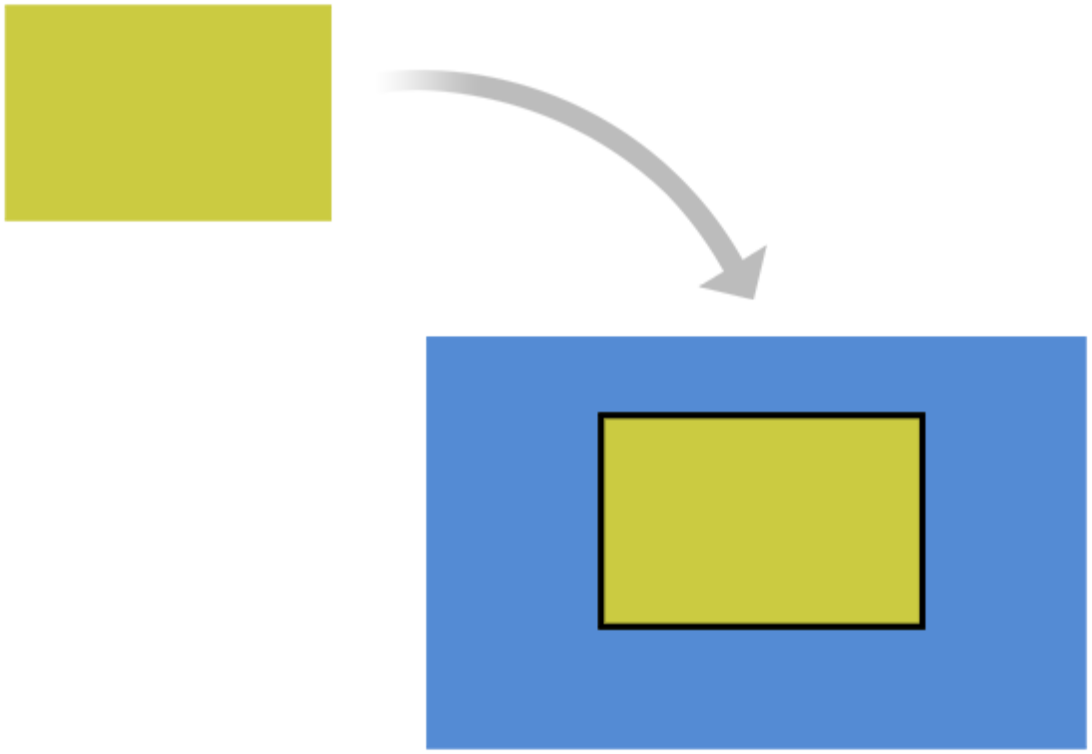
- Incorporation
- Linking
- Modification
- Translation

Incorporation

A developer may copy portions of a FOSS component into your software product.

Relevant terms include:

- Integrating
- Merging
- Pasting
- Adapting
- Inserting



Linking

A developer may link or join a FOSS component with your software product.

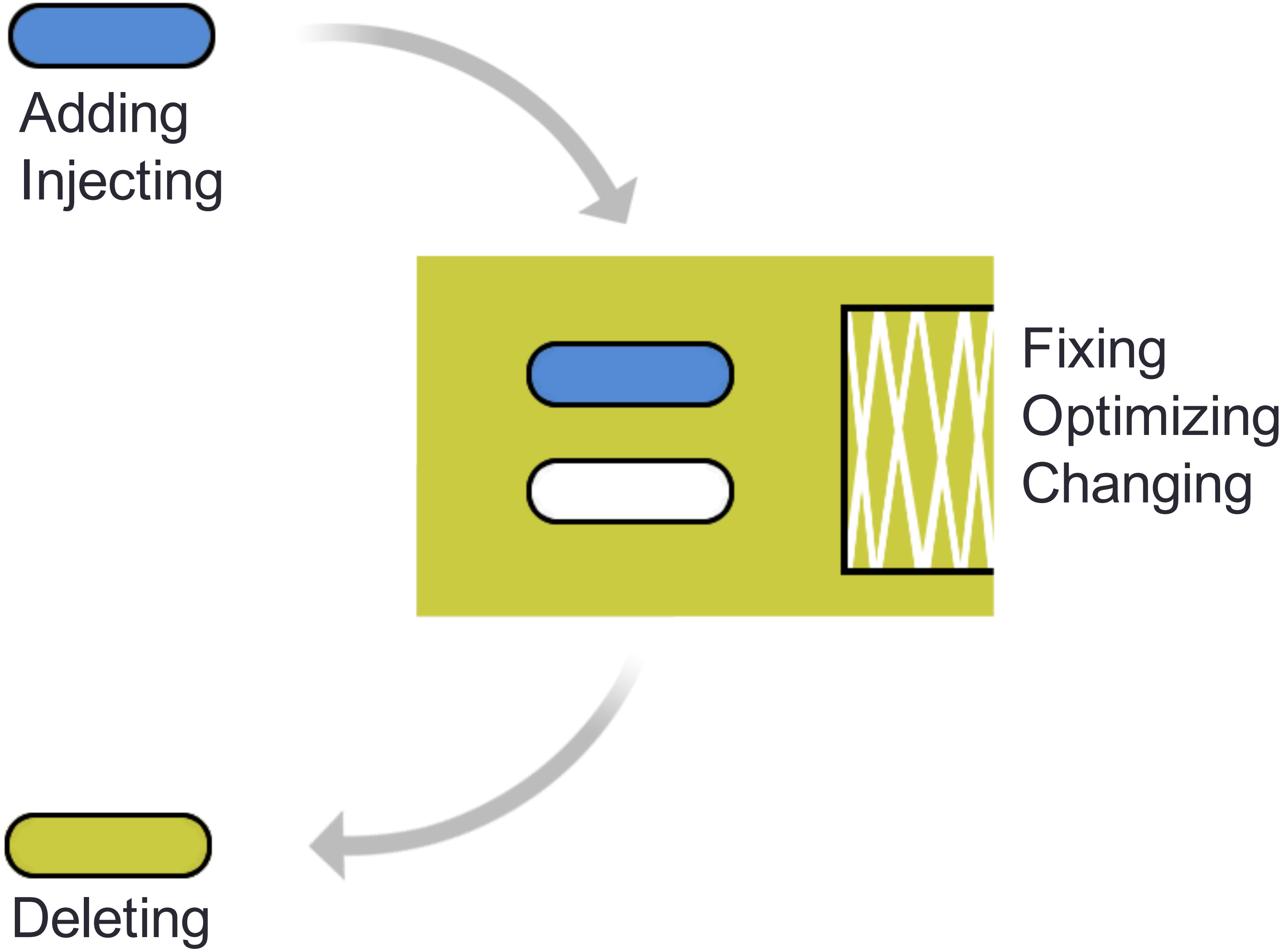
Relevant terms include:

- Static/Dynamic Linking
- Pairing
- Combining
- Utilizing
- Packaging
- Creating interdependency

Modification

A developer may make changes to a FOSS component, including:

- Adding/injecting new code into the FOSS component
- Fixing, optimizing or making changes to the FOSS component
- Deleting or removing code

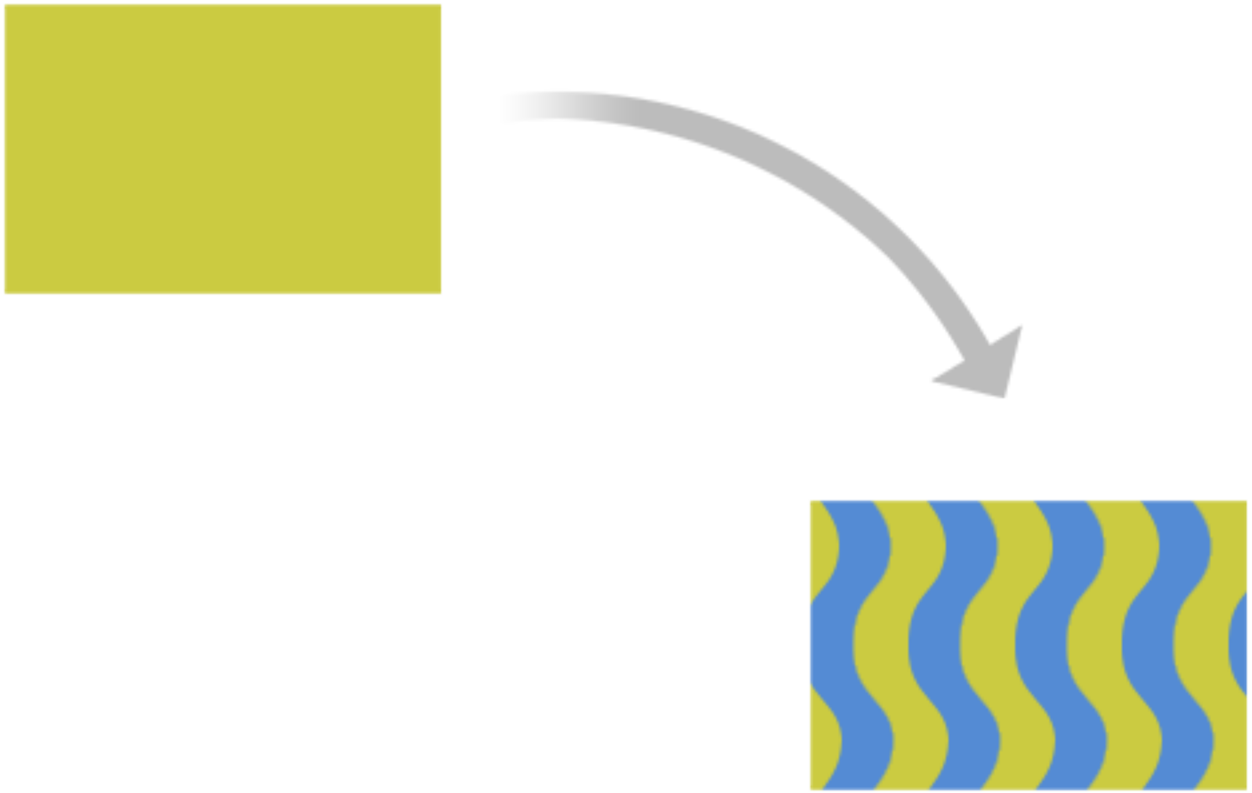


Translation

A developer may transform the code from one state to another.

Examples include:

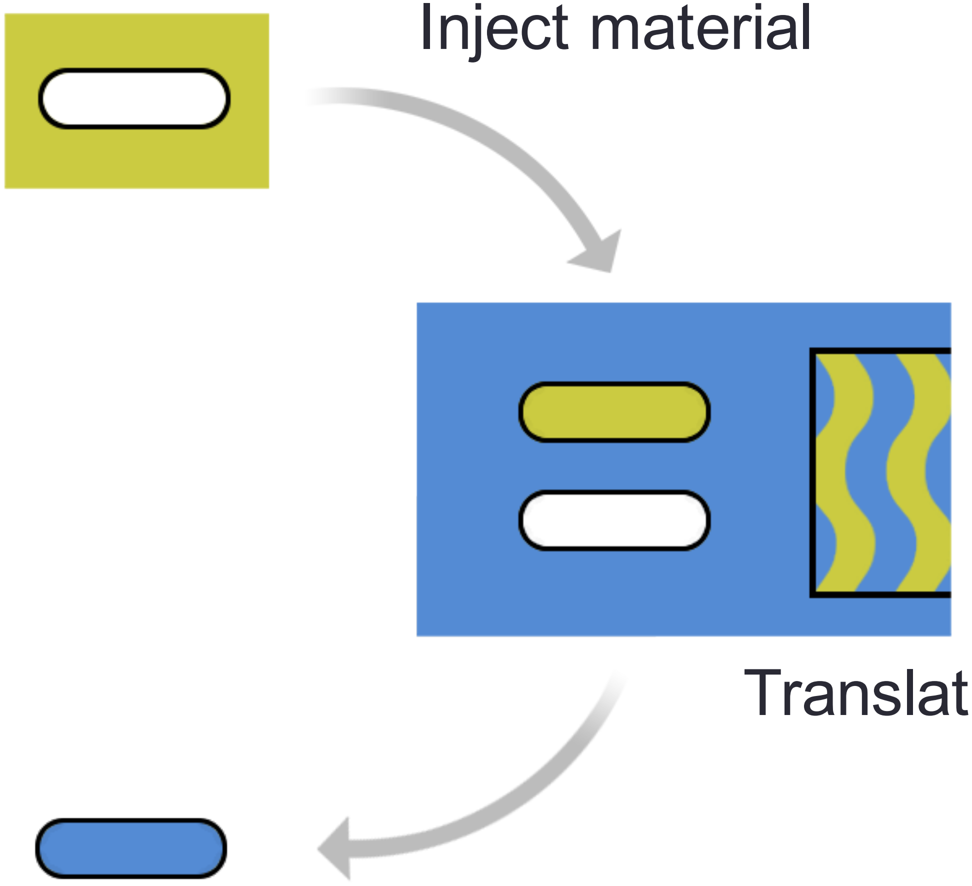
- Translating Chinese to English
- Converting C++ to Java
- Compiling VHDL in a mask or net list
- Compiling into binary



Development Tools

Development tools may perform some of these operations behind the scenes.

For example, a tool may inject portions of its own code into output of the tool.



Inject material

Translate the material

Modify the material

How is the FOSS component distributed?

- Who receives the software?
  - Customer/Partner
  - Community project

- What format for delivery?
  - Source code delivery
  - Binary delivery
  - Pre-loaded onto hardware

Check Your Understanding

- What information is helpful in understanding how software is licensed?
- What information helps identify who is licensing the software?
- What is incorporation?
- What is modification?
- What is linking?
- What is translation?
- What factors are important in assessing a distribution?

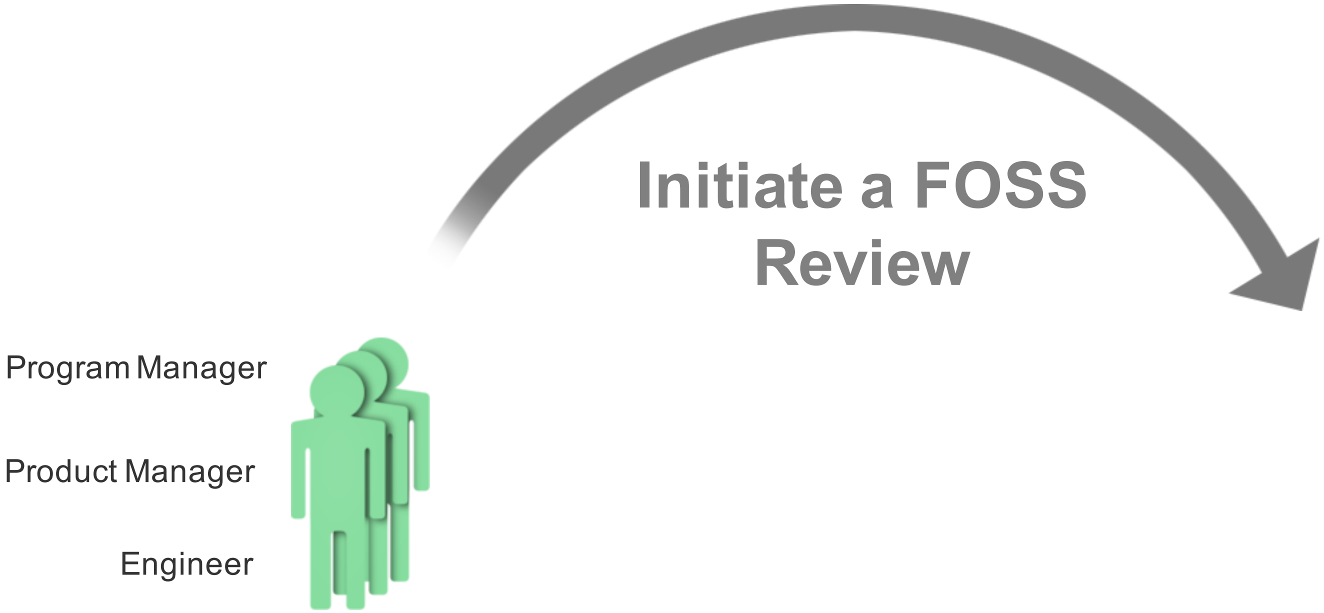
CHAPTER 5

Running a FOSS Review

FOSS Review

- A key element to a FOSS Compliance Program is a *FOSS Review* process, through which a company can analyze and determine its FOSS obligations
- The FOSS Review process includes the following steps:
  - Gather relevant information
  - Analyze and determine license obligations
  - Provide guidance in light of company policy and business objectives

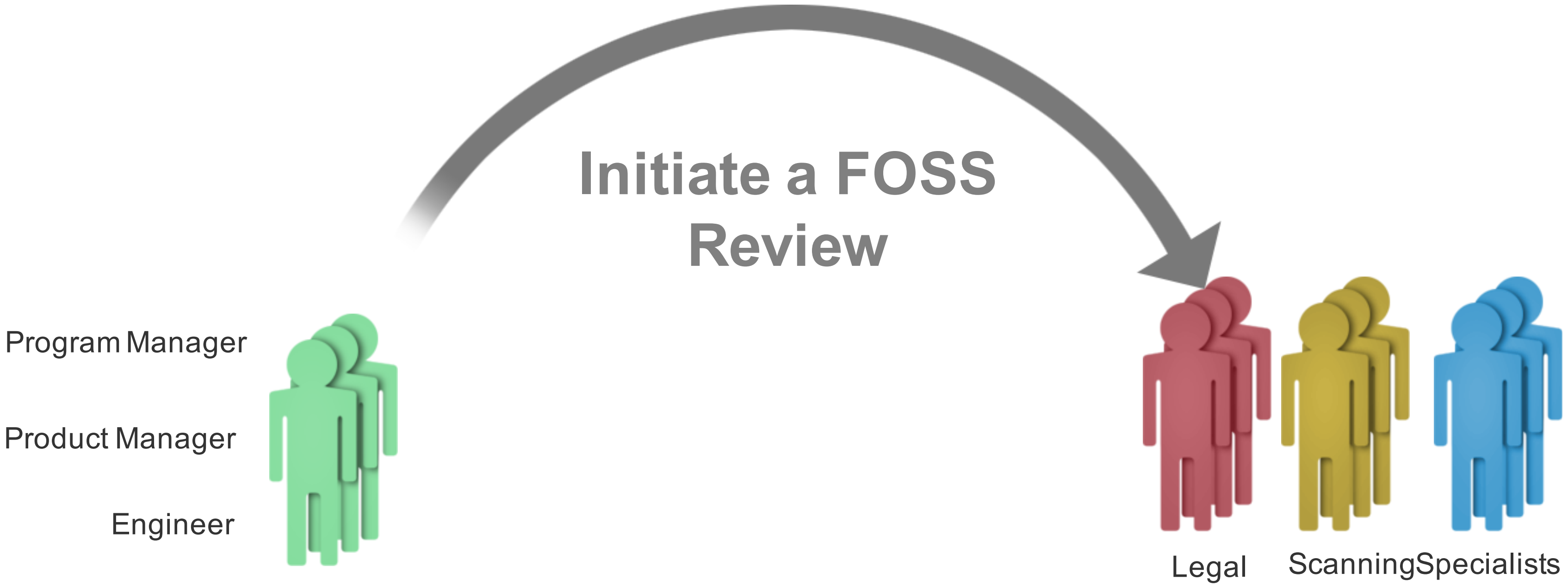
Initiating a FOSS Review



The FOSS Review process should be accessible to Program/Product Managers, Engineers and others who may be working with FOSS.

Note: This process may also start when receiving FOSS-based software from outside vendors.

FOSS Review Team



A FOSS Review alerts and engages the various support groups that work together to support, guide, coordinate and review the use of FOSS. This team may include:

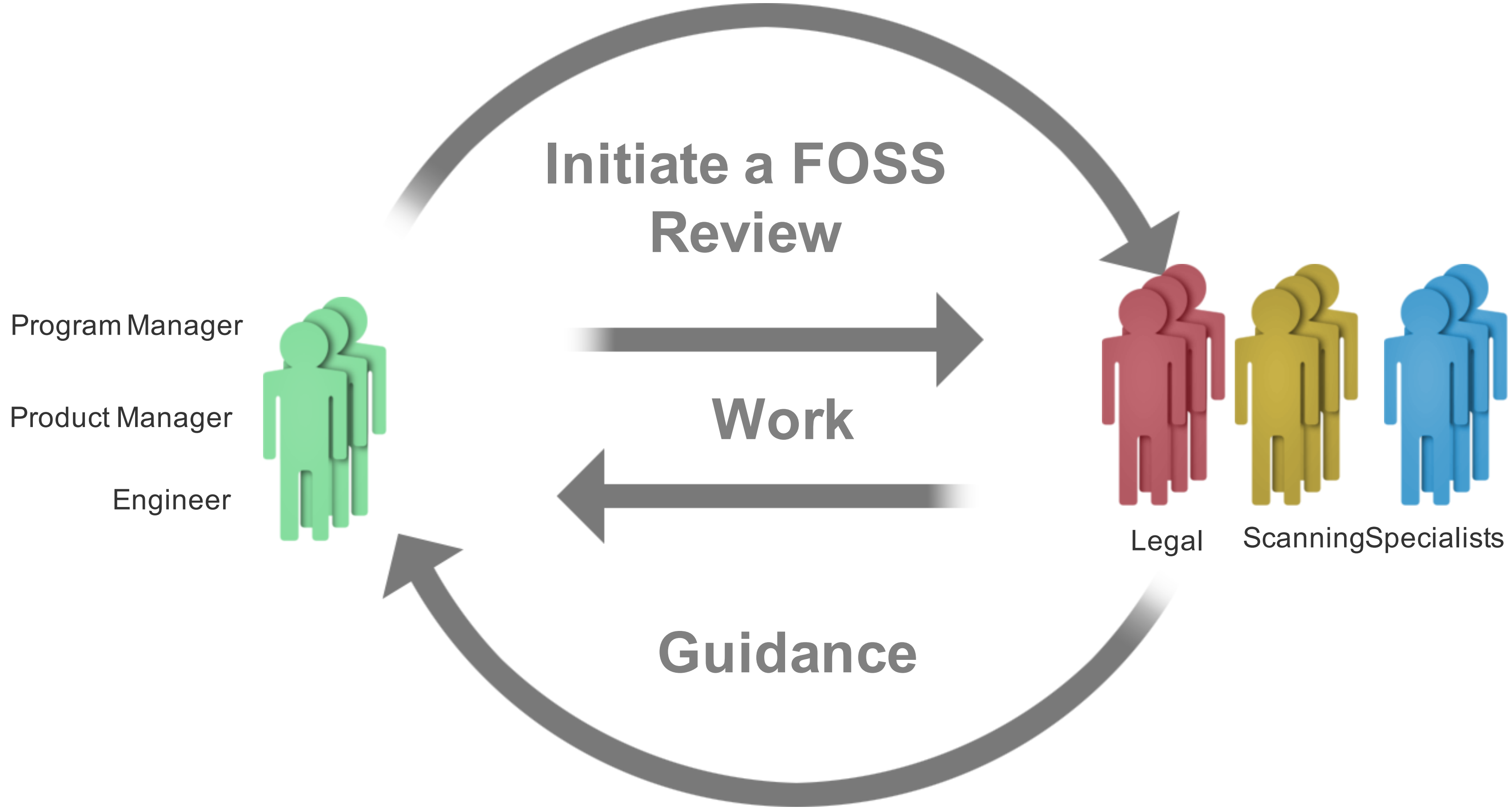
- Legal team to identify and evaluate license obligations
- Scanning and tooling support team to help identify and track FOSS usage
- Specialists working with business interests, commercial licensing, export compliance, etc., who may be impacted by FOSS usage

Analyzing Proposed FOSS Usage

The FOSS Review team should assess the information it has gathered before providing guidance, including for issues such as:

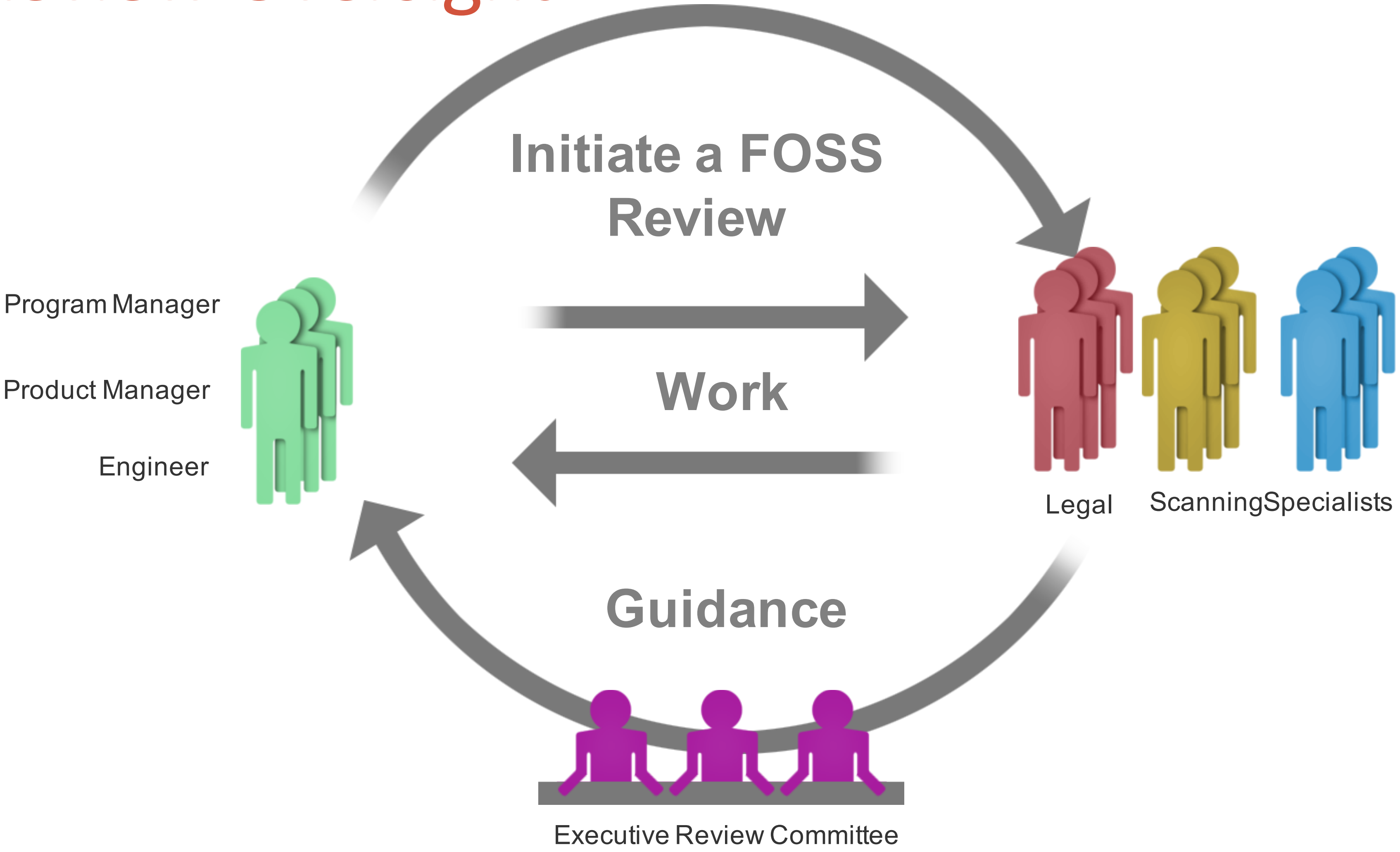
- Completeness, consistency, accuracy (code scanning tools may be used to scan for undisclosed FOSS usage)
- Does the declared license match what is in the code files?
- Does the license truly permit the proposed use of the software?

Working through the FOSS Review



Working through the FOSS Review process is interactive. The work crosses disciplines, including engineering, business and legal teams, and may require in follow-up discussion so that all parties understand the underlying issues. Ultimately, the process should result in clear guidance on FOSS usage.

FOSS Review Oversight



The FOSS Review process should have sufficient oversight in cases of disagreement between any of the parties involved, or when a decision is particularly important.

Check Your Understanding

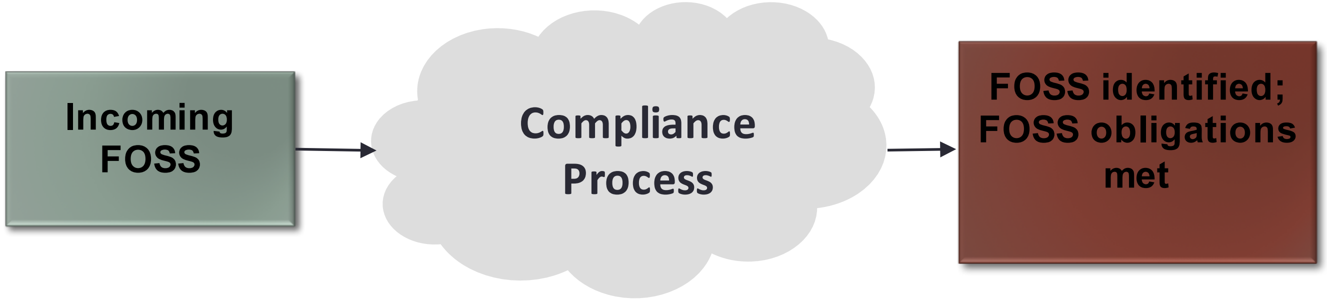
- What is the purpose of a FOSS Review?
- What is the first action you should take if you want to use FOSS components?
- What kinds of information might you collect for a FOSS review?
- What additional information is important when reviewing a FOSS component from an outside vendor?
- What steps can be taken to assess the quality of this information?
- What should you do if you have a question about using FOSS?

CHAPTER 6

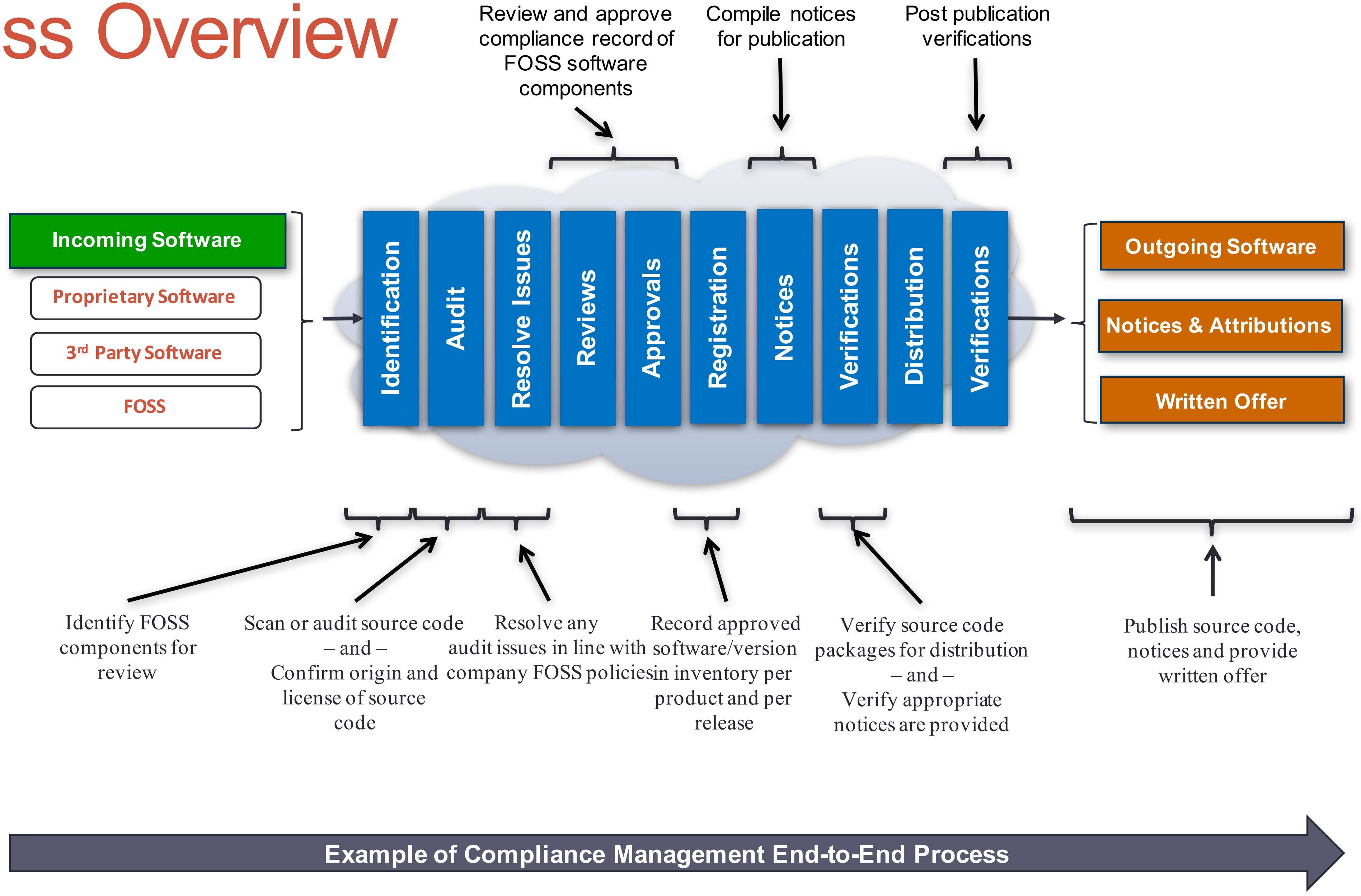
End to End Compliance Management (Example Process)

Introduction

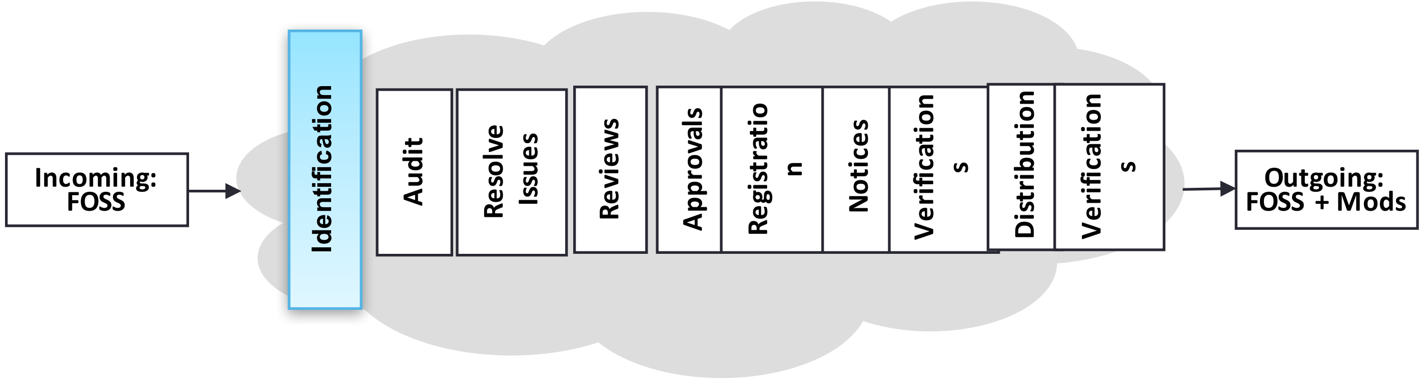
- Compliance management consists of a set of actions that controls the intake and distribution of FOSS used in products (or "Supplied Software" in the OpenChain specification)
- The result of compliance due diligence is an identification of all FOSS used in the Supplied Software and confirmation that all FOSS license obligations have been or will be met
- This chapter provides an example of such a process, and may serve as a resource for forming or improving your internal processes



Process Overview


Review and approve compliance record of FOSS software components
Compile notices for publication
Post publication verifications
Incoming Software
Proprietary Software
3rd Party Software
FOSS
Identification
Audit
Resolve Issues
Reviews
Approvals
Registration
Notices
Verifications
Distribution
Verifications
Outgoing Software
Notices & Attributions
Written Offer
Identify FOSS components for review
Scan or audit source code – and – Confirm origin and license of source code
Resolve any audit issues in line with company FOSS policies
Record approved software/version in inventory per product and per release
Verify source code packages for distribution – and – Verify appropriate notices are provided
Publish source code, notices and provide written offer
Example of Compliance Management End-to-End Process

Identify and Track FOSS Usage



Identify and begin tracking FOSS from all sources

- Pre-requisites:

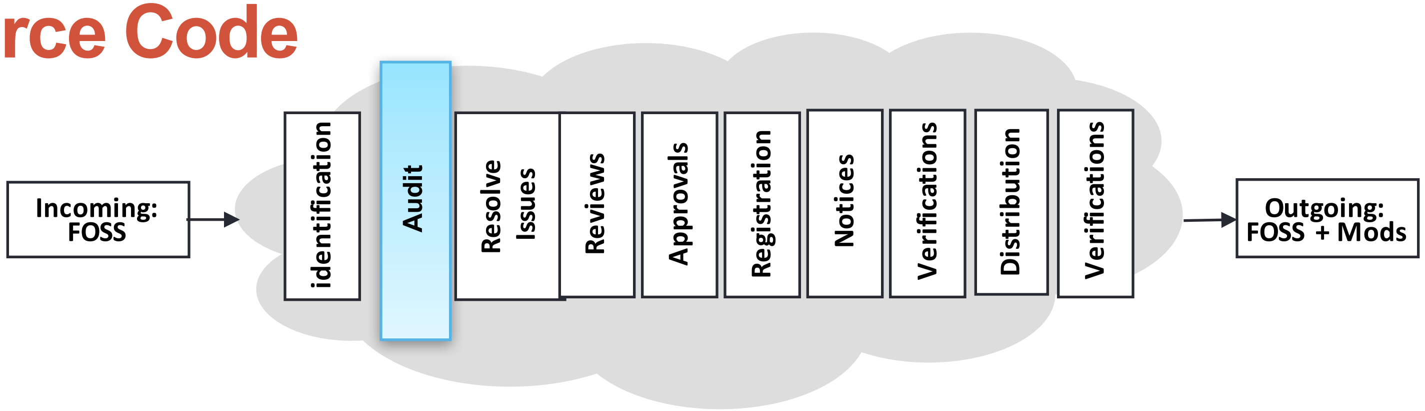
  The process may begin with one of these events:

  - The development team requests the review of a FOSS component or an outgoing release
  - Discovery of FOSS being used without proper authorization
  - Discovery of FOSS being used as part of third party software

- Steps:
  - Incoming requests are recorded
  - Scans of entire platform may be performed
  - Due diligence on any 3rd party provided software
  - Recognize and review any FOSS components added to a repository without an incoming request

- Outcome:
  - A compliance record is created (or updated) for the FOSS
  - An audit is requested to scan or review the source code

Auditing Source Code



Identify FOSS components and their origin and licenses

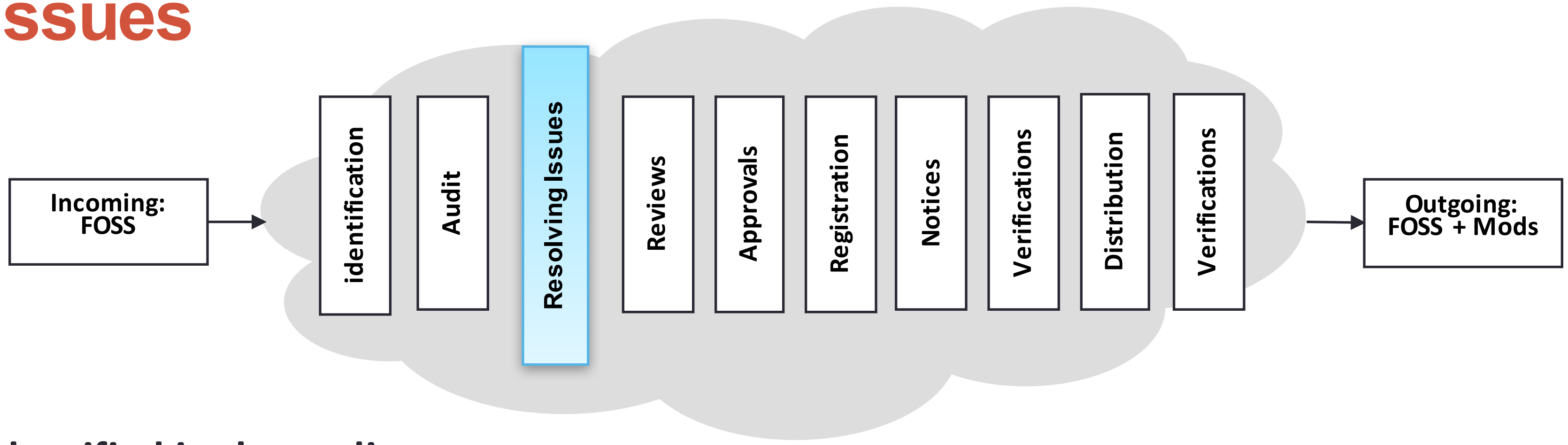
- Pre-requisites:
  - Development team provides a compliance record with information about the FOSS usage
  - In cases where no record is provided by the development team, a record can be created when the FOSS component is discovered

- Steps:
  - Source code for the audit is identified
  - Source may be scanned by a software tool
  - "Hits" from the audit or scan are reviewed and verified as to the proper origin of the code
  - Audits or scans are performed iteratively based on the software development and release lifecycles

- Outcome:

  An audit report identifying the origins and licenses of the source code

Resolving Issues



Resolve all issues identified in the audit

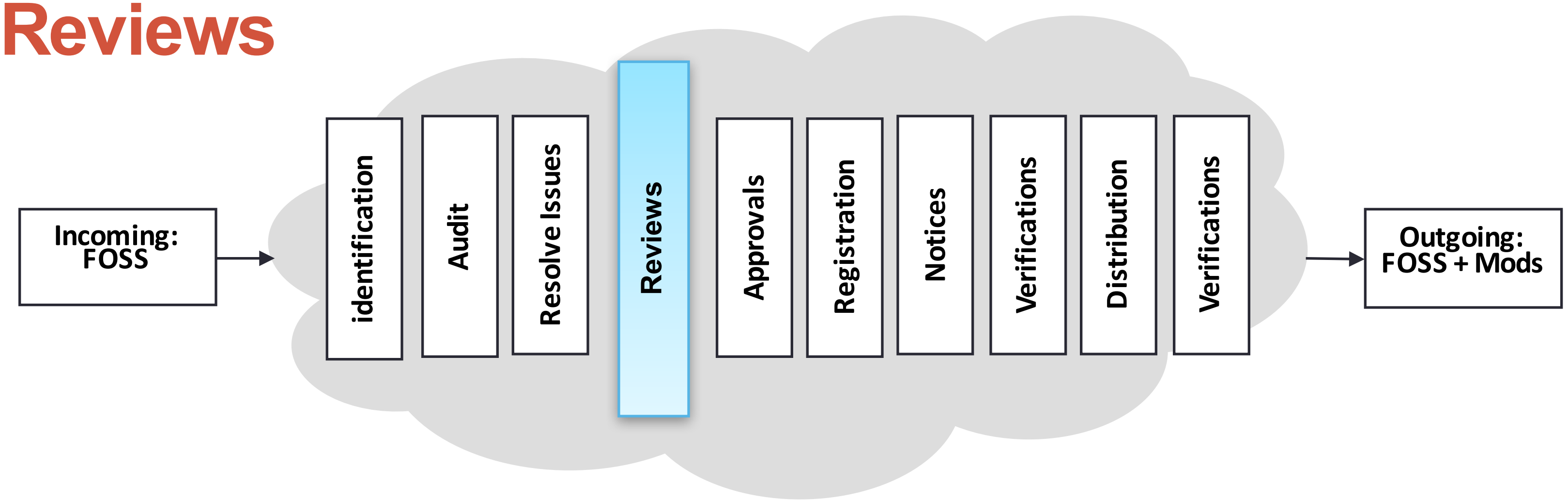
- Pre-requisites:
  - A source code audit or scan has been completed
  - An audit report identifies the origins and licenses of the source code and flags files that need further investigation

- Steps:
  - Provide feedback to the appropriate engineers to resolve issues in the audit report that conflict with your FOSS policy
  - Follow up with engineers to confirm that the issues are resolved

- Outcome:

  A resolution for each of the flagged files in the report and a resolution for any flagged license conflict

Performing Reviews



Review the audit report and confirm any discovered issues are resolved

Pre-requisites:

- Source code has been audited
- All identified issues have been resolved

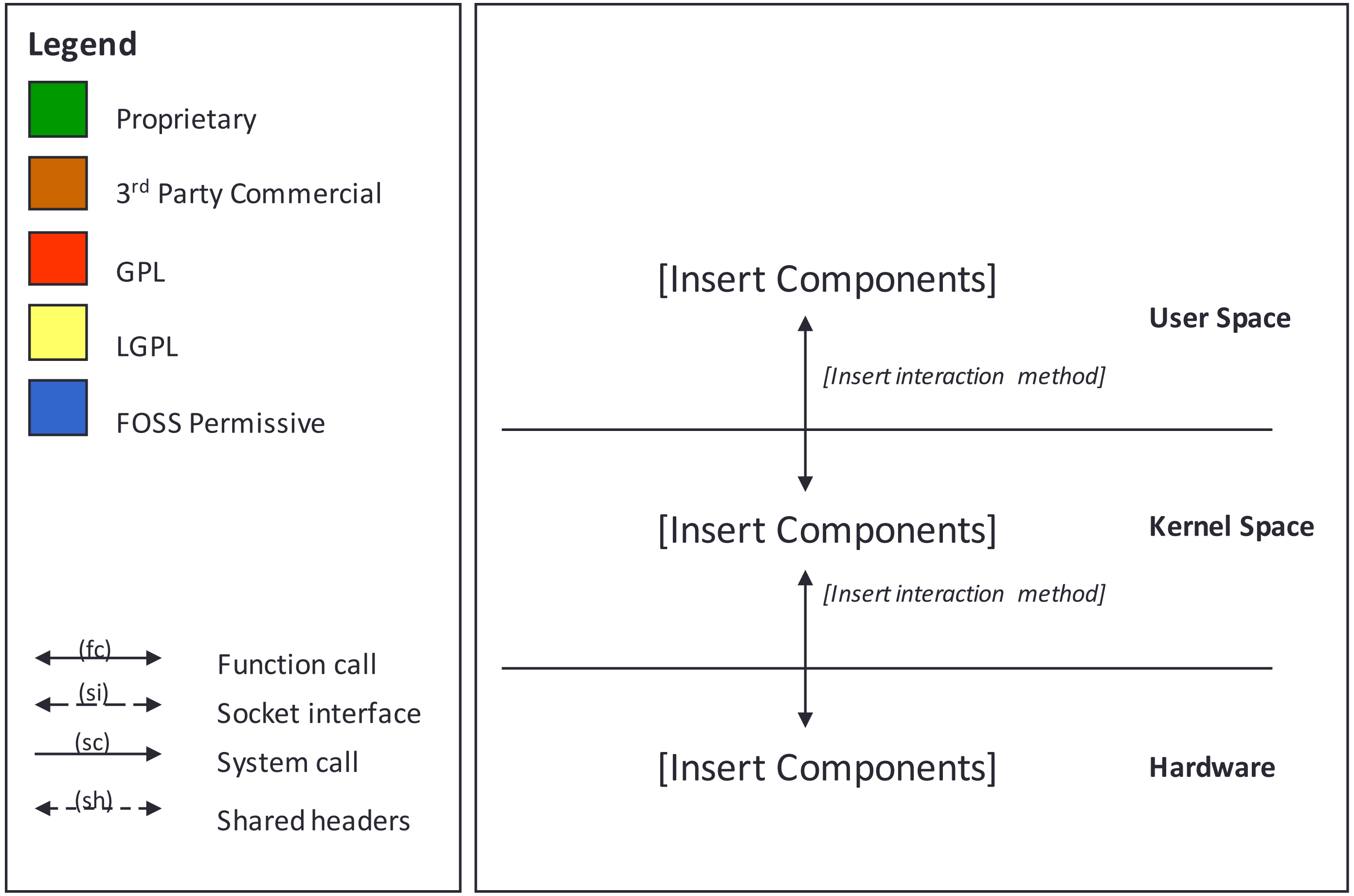
Steps:

- Include appropriate authority levels in review staff
- Conduct FOSS Reviews on audited source code, review software architecture and FOSS usage (see next slide for template)
- Identify obligations under FOSS licenses

Outcome:

- Ensure the software in the audit report conforms with FOSS policies
- Preserve audit report findings and mark resolved issues as ready for the next step (i.e. Approval)

Architecture Review (Example Template)



Legend

🟩 Proprietary

🟧 3rd Party Commercial

🟥 GPL

🟨 LGPL

🟦 FOSS Permissive

(fc) Function call

(si) Socket interface

(sc) System call

(sh) Shared headers

[Insert Components]

[Insert interaction method]

User Space

[Insert Components]

[Insert interaction method]

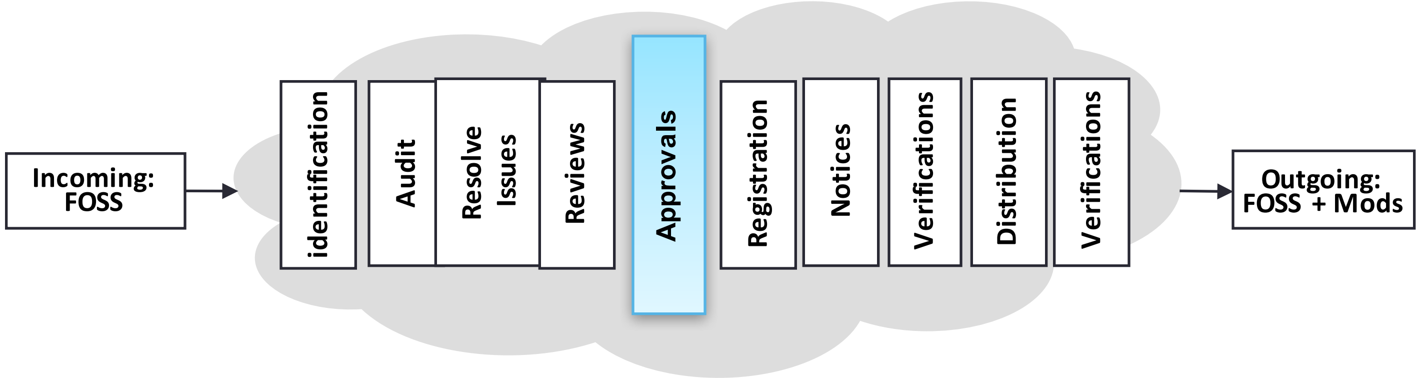
Kernel Space

[Insert Components]

Hardware

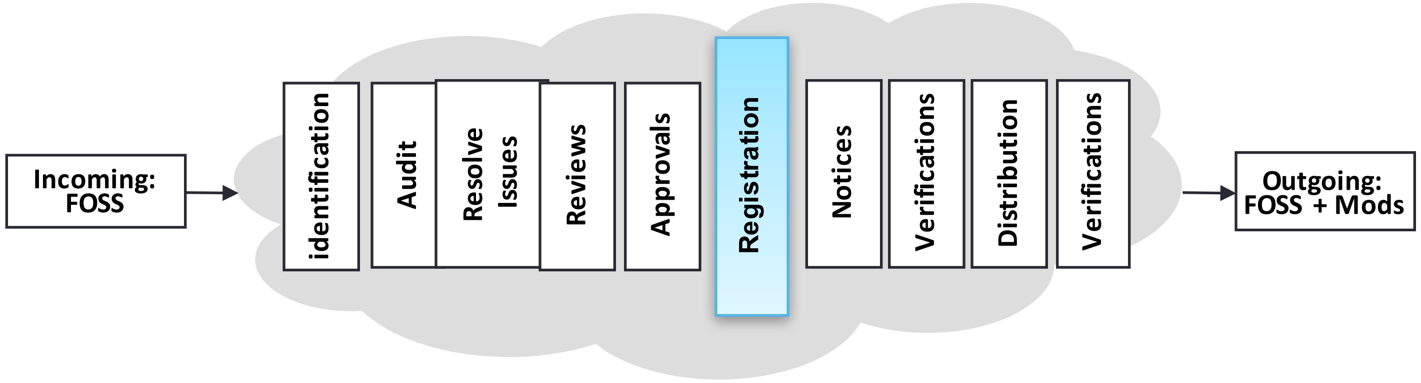
Approvals

- Based on the results of the software audit and review in previous steps, software may or may not be approved for use
- The approval should specify versions of approved FOSS components, the approved usage model for the component, and any other applicable obligations under the FOSS license
- Approvals should be made at appropriate authority levels

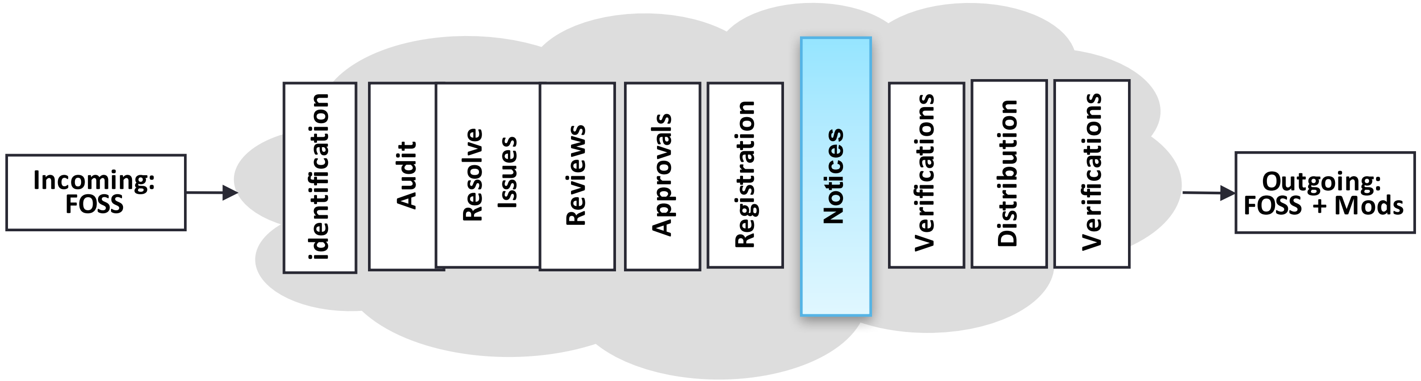


Registration / Approval Tracking

- Once a FOSS component has been approved for usage in a product, it should be added to the software inventory for that product
- The approval and its conditions should be registered in a tracking system
- The tracking system should make it clear that a new approval is needed for a new version of a FOSS component or if a new usage model is proposed

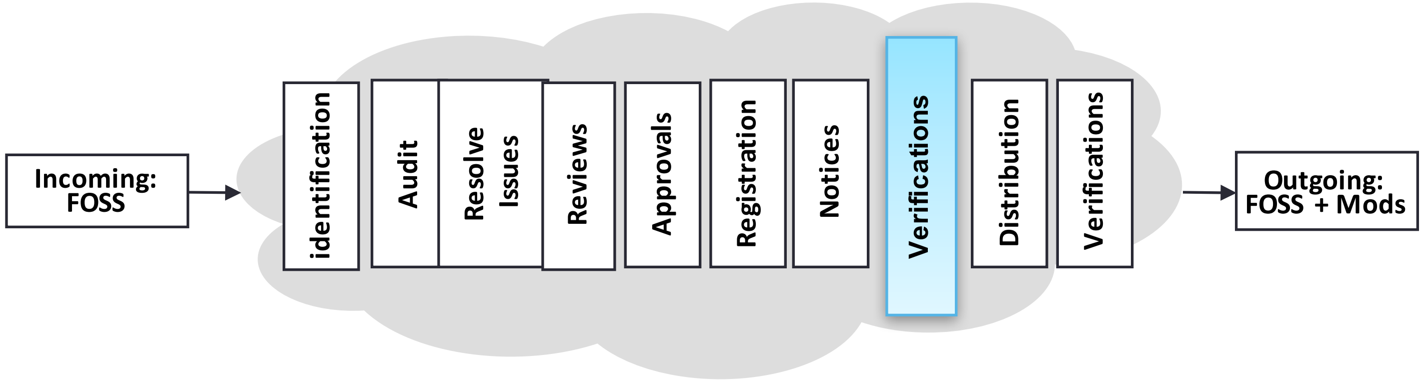


Notices



- **Prepare appropriate notices for any FOSS used in a product release:**
  - Acknowledge the use of FOSS by providing full copyright and attribution notices
  - Inform the end user of their product on how to obtain a copy of the FOSS source code (when applicable, for example in the case of GPL and LGPL)
  - Reproduce the entire text of the license agreements for the FOSS code included in the product as needed

Pre-Distribution Verifications



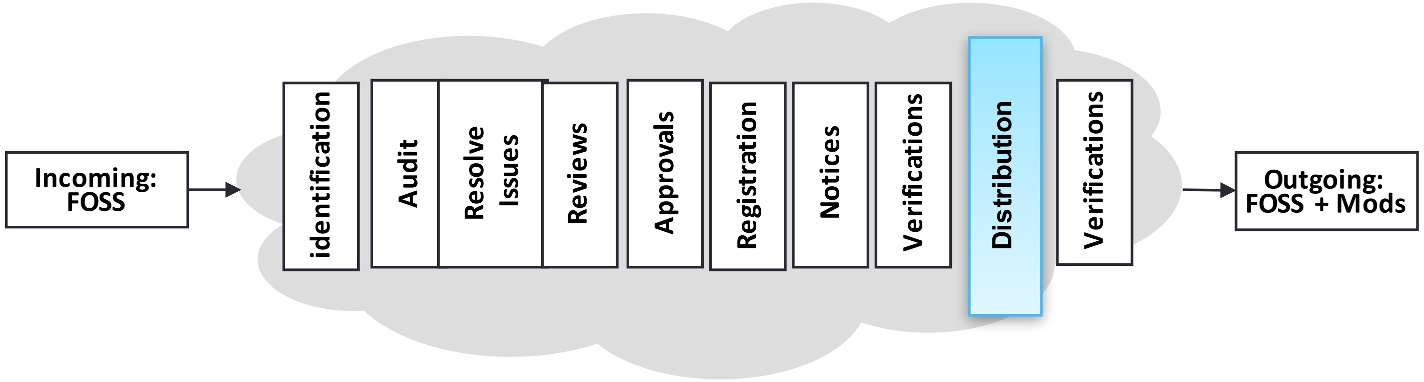
Verify that distributed software has been reviewed and approved

- Pre-requisites:
  - FOSS component has been approved for usage
  - FOSS component has been registered in the software inventory for the release
  - Appropriate notices have been prepared

- Steps:
  - Verify FOSS packages destined for distribution have been identified and approved
  - Verify the reviewed source code matches the binary equivalents shipping in the product
  - Verify all appropriate notices have been included to inform end-users of their right to request source code for identified FOSS
  - Verify compliance with other identified obligations

- Outcome:
  - The distribution package contains only software that has been reviewed and approved
  - "Distributed Compliance Artifacts" (as defined in the OpenChain specification), including appropriate notice files are included in the distribution package or other delivery method

Accompanying Source Code Distribution



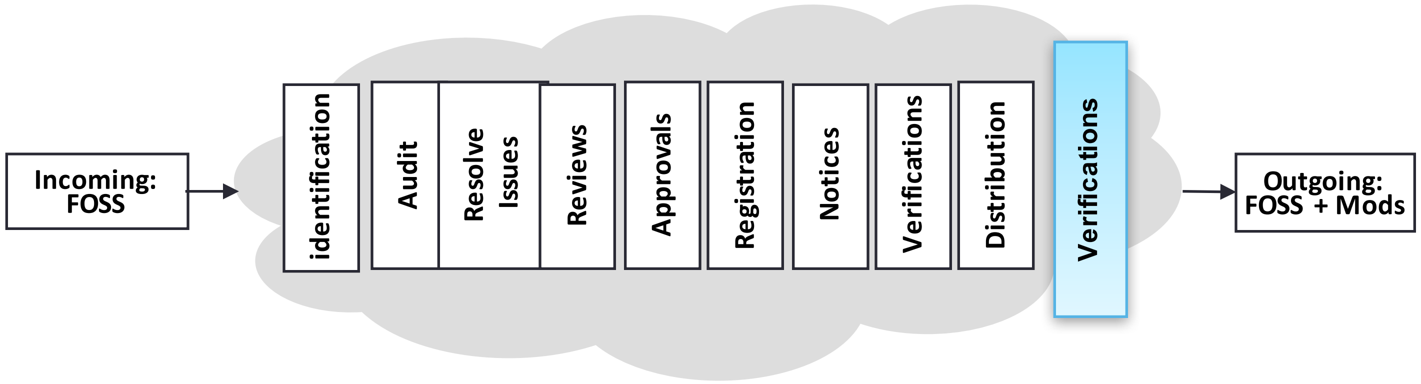
Provide accompanying source code as required

- Pre-requisites:
  - All pre-distribution verification has been completed and no issue is discovered

- Steps:
  - Provide accompanying source code along with any associated build tools and documentation (e.g., by uploading to a distribution website or including in the distribution package)
  - Accompanying source code is identified with labels as to which product and version to which it corresponds

- Outcome:
  - Obligations to provide accompanying source code are met

Final Verifications



Validate compliance with license obligations

- Pre-requisites:
  - Accompanying source code is provided as may be required
  - Appropriate notices have been prepared

- Steps:
  - Verify accompanying source code (if any) has been uploaded or distributed correctly
  - Verify uploaded or distributed source code corresponds to the same version that was approved
  - Verify notices have been properly published and made available
  - Verify other identified obligations are met

- Outcome:
  - Verified Distributed Compliance Artifacts are appropriately provided

Check Your Understanding

- What is involved in compliance due diligence (describe the steps at a high level)?
- What types of issues may need to be resolved as part of compliance management?
- Who should be involved in reviewing audit results?
- What does an architecture review look for?
- What should be included in the FOSS Notices?
- What needs to be distributed for code used under a copyleft license?

CHAPTER 7

Avoiding Compliance Pitfalls

Compliance Pitfalls

This chapter will describe some potential pitfalls to avoid in the compliance process:

1. Intellectual Property (IP) pitfalls
2. License Compliance pitfalls
3. Compliance Process pitfalls

Intellectual Property Pitfalls

Type & Description	Discovery	Avoidance
Unplanned inclusion of copyleft FOSS into proprietary or 3rd party code: This type of failure occurs during the development process when engineers add FOSS code into proprietary or 3rd party source code in the form of copy and paste into proprietary or 3rd party Software in conflict with your FOSS policies.	This type of failure can be discovered by scanning or auditing the source code for possible matches with: • FOSS source code • Copyright notices Automated source code scanning tools may be used for this purpose	This type of failure can be avoided by: • Offering training to engineering staff to bring awareness to compliance issues and to the different types and categories of FOSS licenses and the implications of including FOSS source code in proprietary source code • Conducting regular source code scans or audits for all the source code in the build environment (proprietary, 3rd party and FOSS)

Intellectual Property Pitfalls

Type & Description	Discovery	Avoidance
Unplanned linking of copyleft FOSS into proprietary source code (or vice versa): This type of failure occurs as a result of linking software (FOSS, proprietary, 3rd party) that have conflicting or incompatible licenses	This type of failure can be discovered using the dependency tracking tool that allows you to discover linkages between different software components.	This type of failure can be avoided by: 1. Offering training to engineering staff to avoid linking software components with licenses that conflict with you FOSS policies 2. Continuously running the dependency tracking tool over your build environment
Inclusion of proprietary code into copyleft FOSS through source code modifications	This type of failure can be discovered using the audits or scans to identify and analyze the source code you introduced to the FOSS component.	This type of failures can be avoided by: 1. Offering training to engineering staff 2. Conducting regular code audits

License Compliance Pitfalls

Type & Description	Avoidance
Failure to Provide Accompanying Source Code	This type of failure can be avoided by making source code publishing a checklist item in the product release cycle before the product becomes available in the market place.
Providing the Incorrect Version of Accompanying Source Code	This type of failure can be avoided by adding a verification step into the compliance process to ensure that the accompanying source code for the binary version is being published.
Failure to Publish Accompanying Source Code for FOSS Component Modifications	This type of failure can be avoided by adding a verification step into the compliance process to ensure that source code for modifications are published, rather than only the original source code for the FOSS component

License Compliance Pitfalls

Type & Description	Avoidance
Failure to mark FOSS Source Code Modifications: Failure to mark FOSS source code that has been changed or failure to include a description of the changes.	This type of failure can be avoided by: 1. Adding source code modification marking as a verification step before releasing the source code 2. Offering training to engineering staff to ensure they update copyright markings or license information of all FOSS or proprietary software that is going to be released to the public

Compliance Process Failures

Description	Avoidance	Prevention
Failure by developers to seek approval to use FOSS	This type of failure can be avoided by offering training to Engineering staff on the company's FOSS policies and processes.	This type of failure can be prevented by: 1. Conducting periodic full scan for the software platform to detect any "undeclared" FOSS usage 2. Offering training to engineering staff on the company's FOSS policies and processes 3. Including compliance in the employees performance review
Failure to take the FOSS training	This type of failure can be avoided by ensuring that the completion of the FOSS training is part of the employee's professional development plan and it is monitored for completion as part of the performance review	This type of failure can be prevented by mandating engineering staff to take the FOSS training by a specific date

Compliance Process Failures

Description	Avoidance	Prevention
Failure to audit the source code	This type of failure can be avoided by: 1. Conducting periodic source code scans/audits 2. Ensuring that auditing is a milestone in the iterative development process	This type of failure can be prevented by: 1. Providing proper staffing as to not fall behind in schedule 2. Enforcing periodic audits
Failure to resolve the audit findings (analyzing the "hits" reported by a scan tool or audit)	This type of failure can be avoided by not allowing a compliance ticket to be resolved (i.e. closed) if the audit report is not finalized.	This type of failure can be prevented by implementing blocks in approvals in the FOSS compliance process
Failure to seek review of FOSS in a timely manner	This type of failure can be avoided by initiating FOSS Review requests early even if engineering did not yet decide on the adoption of the FOSS source code	This type of failure can be prevented through education

Ensure Compliance Prior to Product Shipment

- Companies must make compliance a priority before any product (in whatever form) ships
- Prioritizing compliance promotes:
  - More effective use of FOSS within your organization
  - Better relations with the FOSS community and FOSS organizations

Establishing Community Relationships

OPENCHAIN

As a company that uses FOSS in commercial product, it is best to create and maintain a good relationship with the FOSS community, in particular, the specific communities related to the FOSS projects you use and deploy in your commercial product.

In addition, good relationships with FOSS organizations can be very helpful in advising on best way to be compliant and also help out if you experience a compliance issue.

Good relationships with the software communities may also be helpful for two-way communication: upstreaming improvements and getting support from the software developers.

THE LINUX FOUNDATION

Check Your Understanding

- What types of pitfalls can occur in FOSS compliance?
- Give an example of an intellectual property failure.
- Give an example of a license compliance failure.
- Give an example of an compliance process failure.
- What are the benefits of prioritizing compliance?
- What are the benefits of maintaining a good community relationship?