カリキュラム

FOSSトレーニング リファレンス スライド OpenChain 仕様書 1.1版対応

本スライドは米国法令に準じています。米国外では法的要求事項が異なる場合がありますのでコンプライアンス トレーニング プログラムで本スライドを使う際にはこの点を考慮する必要があります。

本スライドは法的助言を提供するものではありません。These slides do not contain legal advice

OpenChain カリキュラムのスライドへようこそ。本スライドは、組織内チームに向けたFOSSコンプライアンスについてトレーニングを実施する際に、トレーニングの補助として使い、OpenChainの仕様への適合を達成するために用いることができます。

このスライドは、半日のトレーニング セッションとして提供することができます。また、各章を分割し、個別のモジュールとして提供することも可能です。各章には質問形式の「理解度チェック」のスライドを設けており、回答はスライドのノート欄に記載されています。これらの質問と回答はFOSSコンプライアンスの組織内テストの素材として使うことができます。

Welcome to the OpenChain Curriculum Slides. These slides can be used to help train internal teams about FOSS compliance issues and to conform with the OpenChain Specification.

You can deliver these slides as one half-day training session or you can deliver each chapter as a separate module. Please note that each chapter has “Check Your Understanding” slides with questions and answers in the slide notes. These can be used as the basis for in-house tests for FOSS compliance.

Disclaimer（免責事項）

- 本文書は、The Linux Foundation におけるOpenChain プロジェクトの英文ドキュメント「 Curriculum for OpenChain Specification 1.1」の公式翻訳版となります。ただし、翻訳版と英語版との間で何らかの意味の違いがある場合には、英語版が優先されます。

  また、OpenChain は世界のメンバー企業が参加しているプロジェクトですが、資料の細部について必ずしも各国の法令に対応していない可能性があります。本翻訳版を日本で活用する際には、各企業の法務部門を加えた検討が不可欠です。

- This is an official translation from the OpenChain Project. It has been translated from the original English text. In the event there is confusion between a translation and the English version, The English text shall take precedence.

OpenChain カリキュラムとは？

- OpenChain プロジェクトは、フリー／オープンソース ソフトウェア（以降「FOSS」）コンプライアンス プログラムの中核となるコンポーネントを明確にし、これを共有することを促進するためのプロジェクト
- OpenChainの中核が**仕様書**（Specification）となる。FOSSコンプライアンス プログラムが満たすべき主要要件を明確にし、これを公開している
- OpenChain **カリキュラム**（Curriculum）は、仕様書を下支えするトレーニング教材であり無償で自由に利用できる
- これらのスライドは、企業が仕様書1.2項記載の要件を満たすことを促進する。また、一般的なコンプライアンス教育でも利用できる

詳細は以下： https://www.openchainproject.org

本スライドでは、OpenChain カリキュラムがどういったもので、これらのスライドがどういった目的のためのものかの説明に役立ちます。

--

This slide helps explain what the OpenChain Curriculum and these slides are for.

コンテンツ

OPENCHAIN

1. 知的財産とは何か？
2. FOSSライセンス概論
3. FOSSコンプライアンス概論
4. FOSSレビューにおけるソフトウェアの重要概念
5. FOSSレビューの実施
6. コンプライアンスマネジメントの始めから終わりまで（プロセス例）
7. コンプライアンスでの落とし穴とその回避
8. 開発者向けガイドライン

このスライドは、単発での3時間トレーニング セッション、もしくは短めのセッションに分け章単位で重点を置いたトレーニングとして実施する場合において、その進め方の説明に用います。

This slide is relevant to providing either a single three hour training session or explaining how a series of shorter sessions focused on “per chapter” training will work.

FOSS ポリシー

- <<本スライドは、FOSSポリシーが企業内のどこに置かれているかを周知するためにご使用ください（OpenChain仕様書1.1の1.1.1項）>>

- FOSSポリシーのサンプルは、The Linux Foundationの
  Open Compliance Programのサイトより入手可能：
  https://www.linux.com/publications/generic-foss-policy

このスライドで、企業が、社内文書として内部FOSSポリシーがどこにあるかを示すことができるようにしていま
す。

This slide is intended to help a company identify where their internal FOSS policy is located in the company documentation.

第1章

知的財産とは何か？

"知的財産”とは何か？

- **著作権（コピーライト）：作者の独創性のある著作物を保護する**
  - （根底のアイデアではなく）表現を保護
  - ソフトウェア、書物、および類似の著作物
- **特許権（パテント）：新規性、非自明性を持つ有用性のある発明**
  - イノベーションを奨励するための限定的な独占権
- **営業秘密：価値ある機密情報を保護する**
- **商標：（言葉、ロゴ、標語、色などの）製品の出所を識別する標識を保護する**
  - 消費者とブランドを保護；消費者の混乱やブランドの希薄化を回避する

本章では、FOSSコンプライアンスに最も関係する、
著作権と特許権に焦点を当てる

ここにある概説で知的財産のすべての側面を網羅することは意図していません。 ここで意図しているのは、「全体像」の観点から、当カリキュラムで議論するのがFOSSコンプライアンスに最も関係する著作権と特許権だということを確認してもらうことです。

This overview is not intended to cover all aspects of Intellectual Property. It is intended to provide context for the “big picture” and to establish that today we are only discussing copyright and patents, the areas most relevant to FOSS compliance.

ソフトウェアにおける著作権の概念

- 基本ルール：著作権は独創的作品を保護する
- 一般的に著作権は、書物、動画、絵画、音楽、地図などの著作物に適用される
- ソフトウェアは、著作権によって保護される。
  - （特許権で保護される）「機能」ではなく、「表現」（実装の細部における独創性）が保護される
  - バイナリコードとソースコードが含まれる
- その作品の著作権保有者は、自らが創作した作品だけをコントロールでき、他の誰かの独立した創作物はコントロールできない
- 著作者の許可なく複製した場合、著作権侵害が生じる可能性がある

このスライドでは、ソフトウェアの著作権についての〝全体像〟を説明しています。

This slide explains the “big picture” of copyright in software.

ソフトウェアに最も関係する著作権における「権利」

OPENCHAIN

- ソフトウェアを 「複製」する 権利： コピーを作成することができる
- 「派生的著作物（Derivative work）※」 を作る権利：修正を加えることができる
  - 派生的著作物(Derivative work)という用語は米国著作権法から来ている
  - 辞書の定義ではなく、法に基づき特定の意味を成す専門的用語（Term of art）
  - 一般的には、原著作物に対し、独自に創造的な作業が十分加えられ、コピー（複製）ではなく独創的な作品であることを示す新たな著作物のことをいう
- 「頒布」する権利
  - 頒布とは、一般的にソフトウェア部品のコピーをバイナリまたはソースコードの形態で、他のエンティティ（個人や外部の企業・組織）に提供する行為とみなされる

注：何をもって「派生的著作物」、「頒布」とするかの解釈はFOSSコミュニティにおいても、関連する法務関係者の間においても議論の対象となっている

※日本の著作権法の「二次的著作物」に該当

このスライドでは、ソフトウェアに対する著作権法の最重要部分を明確にしています。

This slide clarifies the most important parts of copyright law to software.

ソフトウェアにおける特許の概念

- 特許は、機能を保護する：これには、コンピューター プログラムのような演算方法が含まれる
  - 抽象的なアイデアや自然法則は保護しない
- ある国で特許を得ようとする場合、その国での出願することが必須となる。特許が授与された場合、その保有者は、他者の独立した創作であっても、あらゆる人に対しその機能の使用を停止させる権利を持つことになる
- 他者がその技術を使いたい場合、特許ライセンス（その技術の使用、製造、製造委託、販売、販売の提示、および輸入に関する権利※の許諾）を求めることができる
- 他者が同じ発明を独立して創作した場合でも、特許侵害が起こることがある

※それぞれ英文で、rights to「use」,「make」,「have made」,「sell」,「offer for sale」,and「import」

このスライドでは、ソフトウェアに関連する 特許の概念を説明しています。

This slide explains patent concepts relevant to software.

ライセンス

- 「ライセンス」は、著作権や特許の保有者が他者に対し許諾や権利を与える手法
- ライセンスは以下に対し制約を課すことができる
  - 許可される使用形態（商用/非商用、頒布、派生的著作物の作成、製造、製造委託、大量生産）
  - 独占的、または非独占的な許諾条件
  - 地理的な範囲
  - 無期限か、期限付きか
- ライセンスはその許諾に条件を持たせることができる。すなわち何らかの義務を満たした場合にのみ、そのライセンスを得る
  - 例）帰属情報を提供する、互恵的ライセンスを供与する
- 保証、免責、サポート、アップグレード、保守に関する契約事項も含まれる場合がある

この スライドは、「ライセンス」とは何かを説明しています。それは米国法令下の「契約」とは異なっています。ここではライセンスの中にどういったものがあるか、その境界を説明しています。

This slide explains what is a “license.” This is different to a contract under US law. This slides explains the boundaries of what can be in a license.

理解度チェック

- 著作権法はどのようなものを保護しますか？
- ソフトウェアにとって最も重要なのは著作権のどのような権利ですか？
- ソフトウェアは特許の対象になりますか？
- 特許はその保有者に対しどういった権利を付与しますか？
- 単独で自分のソフトウェアを開発した場合でも、そのソフトウェアについて第三者から著作権ライセンスを受ける必要がある可能性がありますか？特許の場合は？

著作権は原作者の独創的な作品を保護します。著作権がアイデアの表現を保護するのに対し、特許は根底にあるアイデアそのものを保護している点で異なります。原作者の作品としては、写真、歌、コンピューターのプログラム コードなどが含まれます。

ソフトウェアの著作権で重要なのは： 複製する権利、派生的著作物を作成する（もしくは改変する）権利、および頒布する権利です。

ソフトウェアは特許を受けることができます。特許はコンピューター プログラムのような演算方法を保護します。ただし、特許は機能を保護するもので、抽象的なアイデアは保護しません。

特許保有者は他者の製品が独立に創出されたかどうかに関係なく、他者が特許を実施することを排除できます。

ソフトウェアを独立して開発した場合、それが独立開発であり、問題とされている著作権付きソフトウェア コードにアクセスしなかったことを示すことができれば、著作権のライセンスは不要である可能性が高いと考えられます。ただし、誰もがその著作権付きのソフトウェア コードを知っており、あなたがアクセスできたと考えることが合理的だとすると、これを証明することは困難となります。ソフトウェアについて特定の特許が主張する機能を読み取れる場合には、そのソフトウェアが独立開発かどうかには関係なく特許ライセンスが必要となるでしょう。そのような例の1つとしてFFMpeg があります。これはビデオの符号化／復号化のためのコーデック機能を提供するフリー ソフトウェア プロジェクトです。しかしながら、何らかのフォーマットを符号化／復号化をするには特許ライセンスが必要となります。

Copyright protects original works of authorship. It's different than patent in that copyright protects the expression of an idea, whereas patent protects the underlying idea itself. Examples of works of authorship include photographs, songs, and computer code.

Most important copyright concepts for software are: right to reproduce, right to make creative works (or right to modify), and right to distribute.

Software can be subject to a patent. Patent protects method of operation, such as computer program. However, patent

protects functionality, and not abstract ideas.

Patent holder can exclude others from practicing the patent, regardless of whether the others have independently created the product.

If you have independently developed your own software, then you may not need a copyright license if you can show the independent development and you had no access to the copyrighted work in question. This is difficult if the copyrighted work is popular such that it'd be reasonable to assume that you had access. If your software reads on a patent, then you will need a patent license regardless of whether you've independently developed the software. An example of this would be FFMpeg, which is a free software project that provides the codecs for encoding and decoding videos. However, you would still need a patent license to encode and decode a certain format.

第2章

FOSSライセンス概論

FOSSライセンス

OPENCHAIN

- FOSSライセンスは、その定義として改変と再頒布を許容する条件の下でソースコードを入手可能にするものをいう
- FOSSライセンスには、帰属情報の提供や著作権宣言文の保持、もしくはソースコードの入手を書面で申し出ること※に関する条件を有する場合がある
- 代表的なライセンスは、オープンソース イニシアチブ（OSI）がそのFOSS定義（OSD）に基づいて承認した一連のライセンス。OSIが承認したライセンスの全リストは、以下のページを参照：
  http://www.opensource.org/licenses/

※「書面による申し出(Written offer)」といわれる

このスライドでは、FOSSライセンスがどういったことをするかの 「全体像」を提供します。またここでは、FOSSライセンスについてさらに多くを調べるための情報源についても説明しています。

This slide provides the “big picture” about what FOSS licenses do. It also explains a resource where you can find out more about some FOSS licenses

パーミッシブ（寛容）なFOSSライセンス

- パーミッシブなFOSSライセンス ：制約が最も少ないFOSSライセンスについて言及する時に用いられる用語
- 例：3条項BSDライセンス
  - BSDライセンスは、著作権表示と同ライセンスの保証に関する免責事項が維持される限り、いかなる目的においてもソースコードもしくはオブジェクト コードの形態で制限ない再頒布を許容するパーミッシブなライセンスの一例
  - このライセンスは派生製品の宣伝に許可なく貢献者の名前を使用することを制限する条項を含んでいる
- その他の例：MITライセンス、Apache-2.0ライセンス

本スライドでは、FOSSライセンスの最も基本的なタイプであり、ライセンス上の要求が最も少ない「パーミッシブ」FOSSライセンスについて説明しています。最も基本的な要求は 著作権表示を含めることです。パーミッシブ ライセンスは下流の受領者に対しソースコードを入手可能にすることを要求しません。コードの保有者はそのソースコードをFOSSライセンスの下で提供しますが、あなたがそのソースコードを他者に提供することは要求しません。

This slide explains ”permissive” FOSS licenses, the most basic type of FOSS license, which usually have minimal requirements. The most basic requirement is to include a copyright notice. Permissive licenses do not require source code to be made available to downstream recipients. The code owner is providing the source code under the FOSS license, but is not requiring that you provide the source code to others.

ライセンスの互恵性とコピーレフトライセンス

- ライセンスの中には、派生的著作物（同じファイル、同じプログラム、あるいは他のバウンダリにあるソフトウェア）を頒布する場合、その頒布が原著作物と同一の条件であることを求めるものがある
- これは、「コピーレフト」、「互恵的」効果と言及される
- GPL version 2.0よりライセンス互恵性の例：

  「『プログラム』またはその一部を含む著作物、あるいは『プログラム』 かその一部から派生した著作物を頒布あるいは発表する場合には、その 全体をこの契約書の条件に従って第三者へ無償で利用許諾しなければならない。 」
- 互恵性やコピーレフトの条項を組み入れたライセンスとして、GPL、 LGPL 、 AGPL、 MPL、および CDDLのすべてのバージョンが挙げられる

このスライドでは、パーミッシブ ライセンスよりも強い要求事項を持つ、より複雑なタイプのFOSSライセンスとして、互恵性と「コピーレフト」 について説明しています。これらは、「原著作物」と「派生的著作物」を原著作物と同じ条件の下で頒布することを要求します。

This slide explains reciprocity and Copyleft, a more complex type of FOSS license that have additional requirements above permissive licenses. They require distribution of the original work and derivative works under the same terms as the original work.

プロプライエタリライセンス、もしくはクローズド ソース ライセンス

OPENCHAIN

- プロプライエタリ ソフトウェア ライセンス（もしくは商用ライセンス、もしくはEULA）は、ソフトウェアの使用、改変、再頒布のすべて、またはいずれかについての制約を有する
- プロプライエタリ ライセンスは、ベンダーごとの独自性がある – 存在するベンダー数と同じバリエーションのプロプライエタリ ライセンスがあり、それぞれを個別に評価しなければならない
- FOSSの開発者たちは、通常、「プロプライエタリ」という用語をFOSSでない商用のライセンスを言い表す際に用いるが、FOSSライセンスもプロプライエタリ ライセンスも、知的財産をベースにしたものであり、どちらもそのソフトウェア資産にライセンスを付与したもの

このスライドでは、プロプライエタリ ライセンスもしくはクローズド ソース ライセンスについて説明しています。これらのライセンスをFOSSライセンスと比較すると、多くの場合、要件やルールに大きな相違があります。

This slide explains proprietary or closed source licenses. These licenses often have very different requirements and rules compared to FOSS licenses.

その他FOSSではないライセンス

- フリーウェア：プロプライエタリ ライセンスの下で、無料または非常に低いコストで頒布されるソフトウェア
  - ソースコードが入手できるものもあれば、できないものもあり、派生的著作物の作成について、一般的には制限される
  - フリーウェアのソフトウェアは、通常すべての機能が使え（機能制約がない）、制限なく使える（使用日数の制約がない）
  - フリーウェアのソフトウェアは、使用タイプ（個人使用、商業目的、学術目的など）についての制約や、ソフトウェアのコピー、頒布、派生的著作物の作成についての制約を課す
- シェアウェア：基本的に試用を前提に、無料で、期間・機能を限定して使用者に提供されるプロプライエタリ ソフトウェア
  - シェアウェアの目的は、将来の購買者がその有用性を評価できるよう、完全版ライセンスの購入前にプログラムを試用する機会を提供すること
  - 大半の企業は、シェアウェアを非常に警戒する。なぜならシェアウェア ベンダーは、そのソフトウェアが組織内で自由に広まってしまった後で、高額なライセンス料の支払いを迫ることがしばしばあるため

その他のタイプのライセンスも使われます。これらは時としてFOSSと混同されることがありますが、その要求事項は実質的に異なります。フリーウェアおよびシェアウェアのライセンスは、FOSSライセンスと同じもの、もしくは互換性があるものとみなすべきではありません。

There are other types of license used. Sometimes these are confused with FOSS but their requirements are actually different. Freeware or Shareware licensing should not be regarded as the same or compatible with FOSS licensing.

その他FOSSではないライセンス

- 「非商用」：FOSSライセンスの特徴の多くを持つものの、非商用使用に限定するライセンスがある（例：CC-BY-NC）
  - FOSSは、その定義としてソフトウェアの使用領域を限定しない
  - いかなる制約もFOSSたらしめることを妨げる。商用使用の制約も使用領域への制約になる

その他のタイプのライセンスも使われます。これらは時としてFOSSと混同されることがありますが、その要求事項は実質的に異なります。フリーウェアおよびシェアウェアのライセンスは、FOSSライセンスと同じもの、もしくは互換性があるものとみなすべきではありません。

There are other types of license used. Sometimes these are confused with FOSS but their requirements are actually different. Freeware or Shareware licensing should not be regarded as the same or compatible with FOSS licensing.

パブリック ドメイン

- パブリック ドメインという用語は、法令で保護されない知的財産を意味する。したがって、パブリック ドメインのものについては、ライセンスを求めずに誰でも使用できる
- 開発者は自身のソフトウェアに対し「パブリック ドメイン宣言」 を行うことができる
  - 例）「本ソフトウェアのすべてのコードと文書類は著作者によりパブリック ドメインに供されました」
  - パブリック ドメイン宣言は、FOSSライセンスと同じものではない
- パブリック ドメイン宣言とは、開発者がそのソフトウェアに対し保有できるあらゆる知的財産権を放棄もしくは消滅させ、制約なくそのソフトウェアが使用できることを明示する試みだが、この宣言の執行可能性については、FOSSコミュニティにおいて議論の対象となっており、また、法としての有効性も国ごとに異なる
- パブリック ドメイン宣言は、保証免責条項のような他の条項を伴うことも多い。その場合、そのソフトウェアは、パブリック ドメインというより、あるライセンスの下にあるとみなすことができる

このスライドではパブリック ドメインについて説明しています。ソフトウェア作品に対しそれがいかなる制約もないことを意味する公開方法の1つといえます。米国ではパブリック ドメイン ソフトウェアもFOSSに含まれる可能性がありますが、すべての国々がその存在を認識したり、パブリック ドメインの下に原作者であることを放棄したりすることを許容するわけではないことに留意しなければなりません。ドイツがその一例です。

This slide explains public domain, a type of release that means the work is released without any restrictions whatsoever by the authors. In the US public domain software can be included in FOSS code, but it should be noted that not all legal jurisdictions recognize the existence or permit the release of authorship under public domain. Germany is one example.

ライセンスの両立性（互換性）※

OPENCHAIN

- ライセンス両立性（互換性）は、（異なるライセンス間で）ライセンス条項に矛盾がないことを確かなものにするプロセス
- 1つのライセンスが何かすることを要求し、他方のライセンスがそうすることを禁じている場合、それらは矛盾する。 その2つのソフトウェア モジュールの組み合わせがライセンスの下での義務を発動させる場合には、2つのライセンスは両立しない（互換ではない）
  - GPL-2.0とEPL-1.0はそれぞれ、 頒布される「派生的著作物」に対し義務を課している
  - GPL-2.0のモジュールが、EPL-1.0のモジュールに結合（Combine）され、統合されたモジュールが頒布される場合、そのモジュールは；
    - ✓（GPL-2.0によれば）GPL-2.0のみで頒布されなければならないことになる、さらに
    - ✓（EPL-1.0によれば）EPL-1.0のみで頒布されなければならないことになる。
    - ✓頒布者は2つの条件を同時に満足することはできないので、このモジュールは頒布できない
    - ✓上記はライセンスが両立しない1つの例

「派生的著作物」の定義はFOSSコミュニティでもその見解が分かれる傾向にあり、法令上の解釈も国ごとに異なる可能性ある。

※FOSSライセンスに係る「Compatibility」の日本語訳として「両立性」、「互換性」2つの方向性があるため併記した

このスライドではライセンスの両立性（互換性）について説明しています。両立性（互換性）は、どのライセンスが一緒に使用できるかを理解する上での考え方です。FOSSにはお互いに両立（互換）できるもの、できないものがあります。コードやライセンスを選択する際にこれは重要な検討事項となります。

This slide explains license compatibility, the way of understanding what licenses can be used together. Some FOSS licenses are compatible with each other. Some are incompatible. This is an important consideration when choosing code and choosing licenses.

告知／通知／表示

告知／通知／表示（Notice）は、 しばしば著作者やライセンスに関する情報を提供する。たとえばファイル先頭のコメント行文字列などの形がある。また、FOSSライセンスでは、ソースコードや文書類内、またはそれらに添える形で告知／表示することを要求する場合がある。これは著作者の功績を称えたり（帰属情報）、そのソフトウェアが改変されたことを明確にさせたりするためである。

- **著作権表示（Copyright notice）：**その著作物の著作権保有者を世に知らしめるべく、ソフトウェアの複写物に掲載される識別子のこと
  例： Copyright © A. Person (2016).
- **ライセンス告知（License notice）**：その製品に含まれるFOSSのライセンス条項や条件を明記し、知らせる表示
- **帰属表示（Attribution notice）：**出荷製品に含まれる表示であり、製品内のFOSSの原作者、出資者の両方もしくはどちらか一方が誰であるかを知らせる
- **改変告知（Modification notice）：**ファイルのソースコードに対して改変を実施したという告知。たとえばファイルの上部に著作権表示を加える、など

このスライドでは、告知／表示（Notice）について説明しています。これは、（FOSSソースコード）ファイル内のコメント文字列（テキスト）によって、著作者やライセンスについて説明するもので、多くの場合、（FOSSソースコード）ファイルに適用されるライセンスを知る最も重要な方法として認識されています。

This slide explains license compatibility, the way of understanding what licenses can be used together. Some FOSS licenses are compatible with each other. Some are incompatible. This is an important consideration when choosing code and choosing licenses.

マルチライセンス

OPENCHAIN

- マルチライセンスとは、複数の異なるライセンス条件下でソフトウェアを同時に頒布する手法
  - 例：ソフトウェアが「デュアルライセンス」である場合、著作権保有者は各受領者に2つのライセンスのどちらかを選択させることができる
- 注：ライセンサ（ライセンス供与者）が複数のライセンスを課す手法と混同しないこと。そのような場合には、**すべての**ライセンス要求を満たさなければならない

このスライドは マルチライセンスについて説明しています。これは、2つ以上のライセンス条件がソフトウェアに適用される状況です。

結合的（Conjunctive） ＝ 複数のライセンスを適用します。
GPL-2.0 プロジェクトはBSD三条項ライセンス下のコードも含みます。
この状況においては両方の条項を満たさなければいけません。
離接的（Disjunctive） ＝ 複数のオープンソース ライセンスから1つのライセンスを選択します。
Mozilla 3ライセンス（tri-license）
Jetty
Ruby

離接的なライセンスは、 FOSSポリシーを策定する際により深く調査すべき重要な事柄となることがあります。

離接的なライセンスの下では、ライセンスを選択することができます。たとえば、GPLとよりパーミッシブなライセンスが選択肢にあった場合、ライセンスの両立性と要件を十分検討した上で、どちらのライセンスで頒布するかを選択できます。

プロジェクトが離接的なライセンスを設定しても、時として、あなたが利用しようとしたコードには1つのライセンスだけが設定されている場合もあります。おそらく、そのコードの作成者が、この選択をすでに実施してしまっているのかもしれません。使いたくないライセンスが選択されていた場合は、原作品の著作権保有者が誰かを明確にし、そこから直接コードを入手するべきかどうかを検討しなければなりません。

例）
MPL 1.1/GPL 2.0/LGPL 2.1 - -
「本ファイルの内容は Mozilla Public License Version 1.1（「本ライセンス」）の適用を受けます

。本ライセンスに従わない限り本ファイルを使用することはできません。

このファイルの内容は、上記に代えて、GNU General Public License Version 2 以降のライセンス（ 「GPL」ライセンス）、もしくは - GNU Lesser General Public License Version 2.1以降のライセンス（ 「LGPL」ライセンス）の条件に従って使用することも可能です。この場合、このファイルの使用には上記の条項ではなくGPLもしくはLGPL ライセンスの条項が適用されます。

このファイルの他者による使用をGPLもしくはLGPLライセンスの条件によってのみ許可し、MPLによる使用を許可したくない対象者は、上記の条項を削除することでその意思を示し、上記条項をGPLもしくはLGPLライセンスで義務付けられている告知およびその他の条項に置き換えてください。対象者が上記の条項を削除しない場合、受領者はMPLまたはGPLもしくはLGPLライセンスのいずれによってもこのファイルを使用することができます。」

「**デュアル（Dual）**」＝ここで記述したすべての状況で使われうる、混乱を招く用語ですが、通常この用語はOSSライセンスもしくは商用ライセンスの選択に関するビジネスモデルについて言及しています。ビジネスモデルとしてのデュアル ライセンスについての詳細は、こちらを参照してください： http://oss-watch.ac.uk/resources/duallicence2

This slides explains multi-licensing. This is the situation where more than set of license terms can apply to a piece of software.

Conjunctive = Multiple licenses apply
- GPL-2.0 project also includes code under BSD-3-Clause
  - In this situation you have to comply with both sets of license terms

Disjunctive = Choice of one open source license or another
- Mozilla tri-license
- Jetty
- Ruby

Disjunctive licensing may be something important to explore more deeply when creating a FOSS policy.

Under disjunctive licensing you have a choice of licensing, i.e. GPL and a more permissive license option, you may choose which license you are going to distribute under depending on license compatibility, license requirements.
Sometimes a project has a disjunctive licensing situation, but only one license is included in your code – so perhaps the person you got the code from already made this choice. If they choose the license you weren't going to use, now you might have to consider if you should figure out who the original © holder is and get the code directly from them

Example:
MPL 1.1/GPL 2.0/LGPL 2.1 - -
"The contents of this file are subject to the Mozilla Public License Version - 1.1 (the "License"); you may not use this file except in compliance with - the License.
. . .
Alternatively, the contents of this file may be used under the terms of - either the GNU General Public License Version 2 or later (the "GPL"), or - the GNU Lesser General Public License Version 2.1 or later (the "LGPL"), - in which case the provisions of the GPL or the LGPL are applicable instead - of those above.

If you wish to allow use of your version of this file only - under the terms of either the GPL or the LGPL, and not to allow others to - use your version of this file under the terms of the MPL, indicate your - decision by deleting the provisions above and replace them with the notice - and other provisions required by the LGPL or the GPL. If you do not delete - the provisions above, a recipient may use your version of this file under - the terms of any one of the MPL, the GPL or the LGPL. "

"**dual**" = confusing term that may be used for any of these situations, but usually refers to business model of OSS license or commercial license choice
For more on dual-licensing as a business model: http://oss-watch.ac.uk/resources/duallicence2

理解度チェック

OPENCHAIN

- FOSSライセンスとはどういったものでしょうか？
- パーミッシブなFOSSライセンスの典型的な義務としてどういったものがありますか？
- パーミッシブなライセンスの名前をいくつか挙げてください。
- ライセンスの互恵性とはどういったことを意味していますか？
- コピーレフトの形態をとるライセンスの名称をいくつか挙げてください。
- コピーレフト ライセンスの下で使用されるコードについては何が頒布される必要がありますか？
- フリーソフトウェアとシェアウェアはFOSSとみなされますか？
- マルチライセンスとはどういったものでしょうか
- FOSSの告知／表示にはどういった情報がありますか？またそれらはどのように使われますか？

FOSSライセンスは、一般に改変と再頒布を許容する条件の下でソースコードを入手可能にするFOSSソフトウェアのライセンスです。

パーミッシブなライセンスの典型的な義務は、著作権表示と保証免責条項がソフトウェアに含まれることです。多くの場合、当該ライセンスでは許可なく著作者の名前を使用することを明に禁止しています。

パーミッシブなFOSSライセンスの例としてはMIT、BSD、Apacheライセンスがあります。

ライセンスの互恵性は、著作権のあるソフトウェアの派生的著作物が同じライセンスの下で入手できなければならないことを意味しています。その他の言い方として、「遺伝的」、「コピーレフト」、「共用（Share-alike）」、さらには、非難的な意味で「ウィルス性」といったものがあります。

コピーレフト スタイルのライセンスにはGPL、LGPLといったものがあります。

コピーレフト スタイルのライセンスには、多くの場合、ソース入手についての義務が規定されており、プログラムやライブラリのバイナリ版を頒布する場合に、そのバイナリに対応したソースコードを提供することを求めます。ソースコードは同じ版名のものでなくてはならず、内容は頒布するバイナリ版に対応していなくてはいけません。

フリーウェアとシェアウェアはFOSSではありません。フリーウェアもシェアウェアもコストなしに入手可能だとしても、使用者に対しソフトウェアの改変を許容していないことがこの理由です。実際には、 多くのフリーウェアとソフトウェアがプロプライエタリソフトウェアに共通するライセンス上の制約を含んでいます。

マルチライセンスはソフトウェアを複数のライセンスの下で使うことができる手法のことをいいます。たとえば、あるオープンソース ソフトウェアはMITとGPLv2の2つのライセンスで供与することができます。そのようなケースでは、使用者がニーズに合わせてライセンスを自由に選択できます。

FOSSの告知／表示には、著作権保有者を識別し、そのソフトウェアをコントロールするライセンスについての情報を含む場合があります。FOSSの告知／表示が改変について告知を提供する場合もあります。FOSSの告知／表示を帰属告知の目的で、保持、再生成することを求めるライセンスもあります。

FOSS licenses are Free and FOSS Software licenses generally make source code available under terms that allow for modification and redistribution.

Typical obligations of a permissive FOSS license are that the copyright notice and warranty disclaimer are included with the software. Very often, the license would expressly prohibits users from using the author's name without permission.

Examples of permissive FOSS licenses include MIT, BSD, and Apache.

License reciprocity means that the derivative work of the copyrighted work must be made available under the same license. Other names being used include "hereditary", "copyleft", "share-alike", and pejoratively "viral."

Examples of copyleft-style licenses include GPL and LGPL.

Copyleft-style licenses often have source availability obligations, which require you to provide accompanying source code when you distribute a binary version of a program or library. The source code should be of the same version and content that corresponds to the binary version you distribute.

Freeware and Shareware are not FOSS. The reason is that even though freeware and shareware are available without cost, they don't allow the users to make modifications to the software. In fact, many of the freeware and shareware contain similar license restrictions common in proprietary software.

Multi-license refers to the practice where software is made available under multiple licenses. For example, an open source software can be dual-licensed under MIT and GPLv2. In that case, you are free to choose the license that suits your need.

FOSS Notices may include information about the identity of the copyright holders and the license governing the software. FOSS Notices may provide notice about modifications. Some licenses require that FOSS Notices be retained or reproduced for attribution purposes.

第3章

FOSSコンプライアンス概論

FOSSコンプライアンスのゴール

OPENCHAIN

- **自らの義務を認識すること。**自身のソフトウェアに存在するFOSSコンポーネントを特定、追跡するためのプロセスを持つ必要がある

- **使用されるFOSSに対しすべてのライセンス義務を果たすこと。**組織のプロセスは、事業遂行上生じるFOSSライセンスの義務に対応できる必要がある

このスライドではFOSSコンプライアンスには 目的が2つあることを説明しています。1つは、自身の義務（FOSSを検出し、追跡する）を認識し、そこで得た情報を維持するプロセスを持つことです。もう1つは、ライセンスの義務を果たすことです。

This slide explains that FOSS compliance is really a two-part goal. The first is to know your obligations and have a process to support this knowledge. The second is to satisfy the obligations.

履行すべきコンプライアンスの義務には どんなものがあるか？

OPENCHAIN

関与するFOSSライセンスにもよるが、コンプライアンスの義務として以下のようなものがある。

- **帰属やその他告知。**ソースコード、製品の関連文書、ユーザー インターフェースのすべてもしくはいずれかに著作権やライセンスに係る文言を提供し、これを保持することが必要とされる場合がある、これにより下流のユーザーがソフトウェアの起源やライセンスによって認められた権利を知ることができる。また、改変に関する告知や、ライセンス全文を提供する必要があることもある
- **ソースコードの提供。**当該FOSSソフトウェアの、もしくはこれに実施した改変、結合ないしはリンクされたソフトウェアのソースコード、およびビルド処理を制御するスクリプトの提供が必要とされる場合がある
- **互恵的な対応。**改変バージョンもしくは派生的著作物に対してもFOSSコンポーネントと同一のライセンスを維持することが求められる場合がある
- **その他の条件。**著作権保有者の名前や商標の使用について制限、混乱を避けるべく改変バージョンに対し、異なる名前の使用要求、ライセンス違反における解除（Terminate）といったことがFOSSライセンスに伴う場合がある

このスライドでは、代表的なFOSSライセンスにおいてどのようなコンプライス義務を履行しなければならないかについて話を展開しています。

ソースコードの入手可能性についてはFOSSライセンスで定められます。そのFOSSソフトウェアに対してのみ求められることもあれば、本スライドに記載したソフトウェアすべてに求められることもあります。

This slide expands on what compliance obligations must be satisfied in typical FOSS licenses.

The scope of source code availability is determined by the FOSS license. Some licenses may require source code availability for only the FOSS software. Others may require all the software described in the slide.

FOSSコンプライアンスにおける論点：頒布

- 外部に対するマテリアル（バイナリ、ソースコードなど）の配布
  - ユーザー機器やモバイル デバイスにダウンロードされるアプリケーション
  - JavaScript、 Web クライアント、ユーザー機器にダウンロードされるコード など
- いくつかのFOSSライセンスについては、コンピューター ネットワークを通じたアクセスが「トリガー イベント」となりうる
  - いくつかのライセンスで、サーバー上で実行されるソフトウェアへのアクセスを可能にすること（例：Affero GPLのすべての版についてソフトウェアを改変した場合）や、「コンピューター ネットワークを通じユーザーがリモートで当該FOSSと相互に作用する」場合を含めたトリガー イベントを定義している

このスライドでは、いつFOSSライセンスの義務が「発動(trigger)される」のかについて説明しています。FOSS ライセンスは著作権ライセンスであり、基本的なコンプライアンスのトリガーはコードを他の法人（legal entity）に 頒布する時です。

This slide explains when FOSS obligations are “triggered.” FOSS licenses are copyright licenses and the basic compliance trigger is when you distribute code to another legal entity.

FOSSコンプライアンスにおける論点：改変

- 既存プログラムに対する変更（例：ファイル中のコードの追加、削除、コンポーネントを組み合わせる行為）
- いくつかのFOSSライセンスには、改変により頒布の際に以下のような追加義務が生じるものがある
  - 改変の告知を提供すること
  - （製品に対応した）添付ソースコードを提供すること
  - 改変結果をそのFOSSコンポーネントと同じライセンス下で許諾すること

このスライドでは、 コードの改変がFOSSライセンス下の義務を課すものとなりうることを説明しています。また、派生的著作物についても若干触れています。

This slide explains that modifying code can impose obligations under FOSS licenses. It explains a little bit about derivative works.

FOSSコンプライアンス プログラム

FOSSコンプライアンスを成功させてきた組織は （ポリシー、プロセス、トレーニングやツールなどから成る）独自のFOSSコンプライアンス プログラムを作り上げている。それには以下のような意図がある。

1. （商用もしくはそれ以外の） 製品におけるFOSSの効果的使用を促進する
2. FOSS開発者／権利保有者の権利を尊重し、ライセンス義務を果たす
3. FOSSコミュニティに参加し、コントリビュートする

このスライドでは、コンプライアンス プログラムがどのように機能するかについて大まかに（基本的概要として）説明しています。

This slide explains how FOSS compliance programs work in “broad stokes” (a basic overview).

コンプライアンスを実践する

以下対応のためビジネスプロセスおよび十分な数のスタッフを準備する：

- 外部、内製問わずすべてのソフトウェアの起源とライセンスの確認
- 開発プロセスにおけるFOSSソフトウェアの追跡
- FOSSレビューの実施と、ライセンス義務の確認
- 製品出荷時におけるライセンス義務の履行
- FOSSコンプライアンス プログラムの監督、ポリシーの策定、およびコンプライアンスに関わる意思決定
- トレーニング

このスライドでは、FOSSコンプライアンス実務が組織内でどのように機能するかについて詳しく説明しています。

This slide explains more about how FOSS compliance practices can work in an organization.

コンプライアンスのメリット

OPENCHAIN

ロバストなFOSSコンプライアンス プログラムがもたらすメリット：

- FOSSのメリットや、FOSSが組織に与える影響についての理解が深まる
- FOSSの使用に伴うコストとリスクについての理解が深まる
- 有効なFOSSソリューションについての知識が高まる
- コンプライアンス違反のリスクを管理、低減でき、開発者や権利保有者が下したライセンス選択を尊重するようになる
- FOSSコミュニティやFOSS関連組織とより良い関係を育むことができる

このスライドではコンプライアンス がライセンスの法的義務の履行という域を超え、組織にもたらすメリットについて述べています。

This slide describes some of the benefits that compliance brings to an organization beyond the fact of fulfilling the legal obligations of the license.

理解度チェック

- FOSSコンプライアンスとは何を意味しますか？
- FOSSコンプライアンス プログラムの2つの主要なゴールとは何ですか？
- FOSSコンプライアンスプログラムを実践する上で重要なものを挙げ、その内容を述べてください。
- FOSSコンプライアンスプログラムのメリットとしてどんなものがありますか？

FOSSコンプライアンスとは、FOSSのライセンス条項に従うことを意味します。これは、ライセンスについての理解、ライセンス条項を支えるプロセスの具備、見落としや誤りに対処するプロセスの具備といったことを伴います。

FOSSコンプライアンスプログラムの2つの主要なゴールとは、**自身の義務を知ること**と**義務を果たすこと**です。

FOSSコンプライアンス プログラムでの重要な業務には以下が含まれます：
- FOSSソフトウェアの起源とライセンスの確認
- 開発プロセスにおけるFOSSソフトウェアの追跡
- FOSSレビューの実施と、ライセンス義務の確認
- 製品出荷時のライセンス義務の履行
- FOSSコンプライアンス プログラムに対する監督、ポリシーの策定およびコンプライアンスに関わる意思決定
- トレーニング

FOSSコンプライアンス プログラムは、さまざまなメリットを提供します。たとえばFOSSが組織にどう影響を与えるかという点や、FOSSに関連づけられるコストやリスクについての理解の向上、またFOSSコミュニティとのより良い関係、有効なFOSSソリューションについての知識の向上といった点があります。

FOSS compliance means following the licensing terms of FOSS licenses. It involves understanding the licenses, having processes to support the license terms, and having processes to address any oversights or errors.

The two main goals of a FOSS compliance program are **know your obligations** and to **satisfy your obligations**.

The important business practices of a FOSS compliance program include:
- 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

A FOSS compliance program provides various benefits such as an increased understanding of how FOSS impacts your organization, an increased understanding of the costs and risks associated with FOSS, better relations with the FOSS community and increased knowledge of available FOSS solutions.

第4章

FOSSレビューにおける ソフトウェアの重要概念

そのコンポーネントをどう使うのか？

共通するシナリオに含まれるもの：

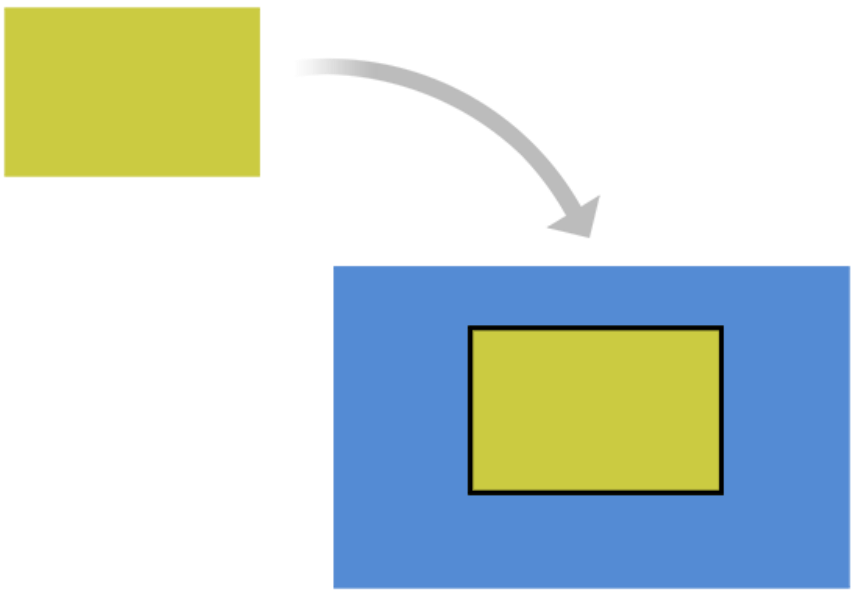
- 取り込む（Incorporation）
- リンクする（Linking）
- 改変する（Modification）
- 翻訳する（Translation）

このスライドは コンプライアンスにおいてFOSSコンポーネントの使用でどういったことを考慮すべきかという点について触れています。ユースケースが異なれば法的効果も違ってきます。次の数枚のスライドでこれらのコンセプトを具体的に説明していきます。

This slide is about how the use of FOSS components is a consideration for your compliance. Different use cases will have different legal effects. The next few slides explain these concepts in more detail.

取り込む（Incorporation）

開発者はFOSSコンポーネントの一部を自身のソフトウェア製品にコピーできる。



関連する用語：

- 統合する（Integrating）
- 結合する（Merging）
- 貼り付ける（Pasting）
- 適応させる（Adapting）
- 挿入する（Inserting）

このスライドでは、FOSSを使う際の「取り込む」の意味について概説しています。

This slides outlines what incorporation means when using FOSS.

リンクする（Linking）

開発者はFOSSコンポーネントを自身のソフトウェア製品とリンクもしくは接合（join） することができる。

関連する用語：

- 静的／動的リンクする（Static/Dynamic Linking）
- 対合する（Pairing）
- 結合する（Combining）
- 活用する（Utilizing）
- パッケージ化する（Packaging）
- 相互依存性を生成する（Creating interdependency）

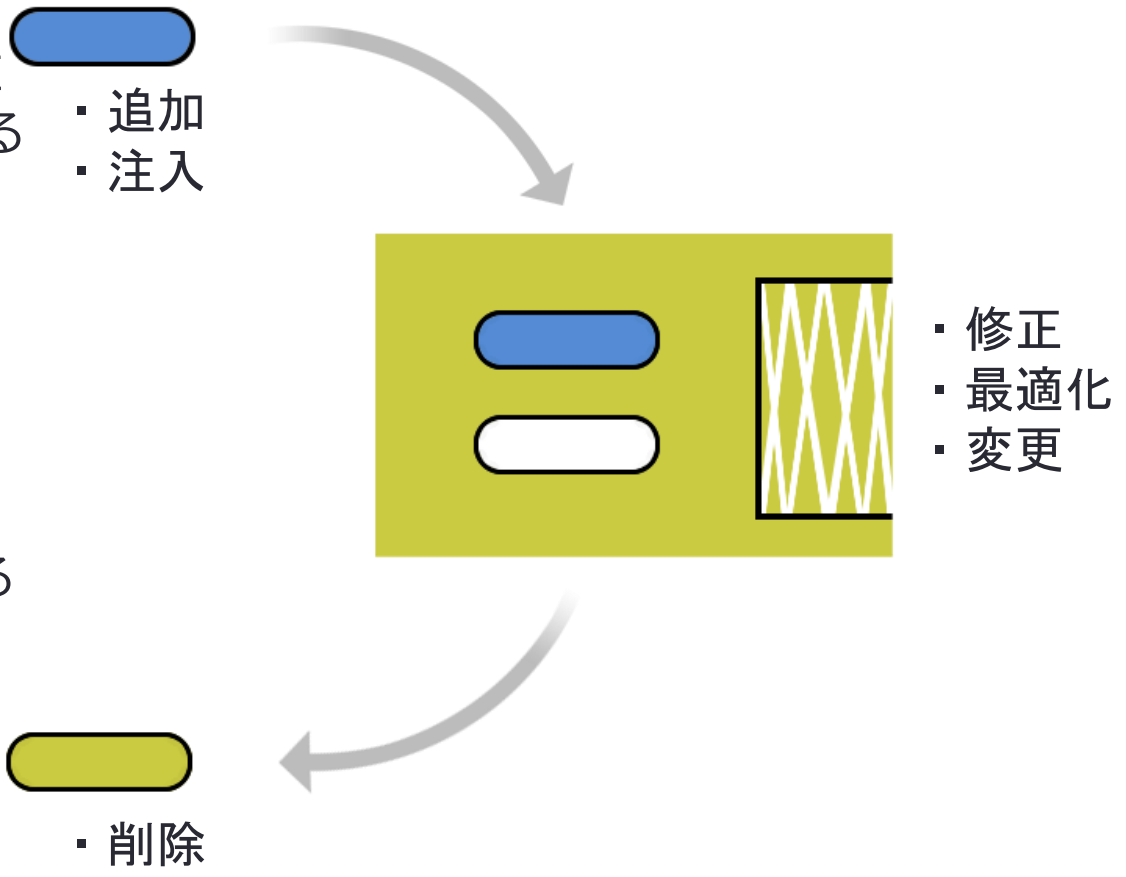
このスライドでは、FOSSを使う際の「リンク」の意味について概説しています。

This slides outlines what linking means when using FOSS.

改変する（Modification）

開発者はFOSSコンポーネントに対して、次のように変更を加えることができる：

- FOSSコンポーネントに新たなコードを追加／注入する（Adding/Injecting）
- FOSSコンポーネントを修正する（Fixing）、最適化する（Optimizing）または変更する（Making change）
- コードを削除する（Deleting）または除去する（Removing）

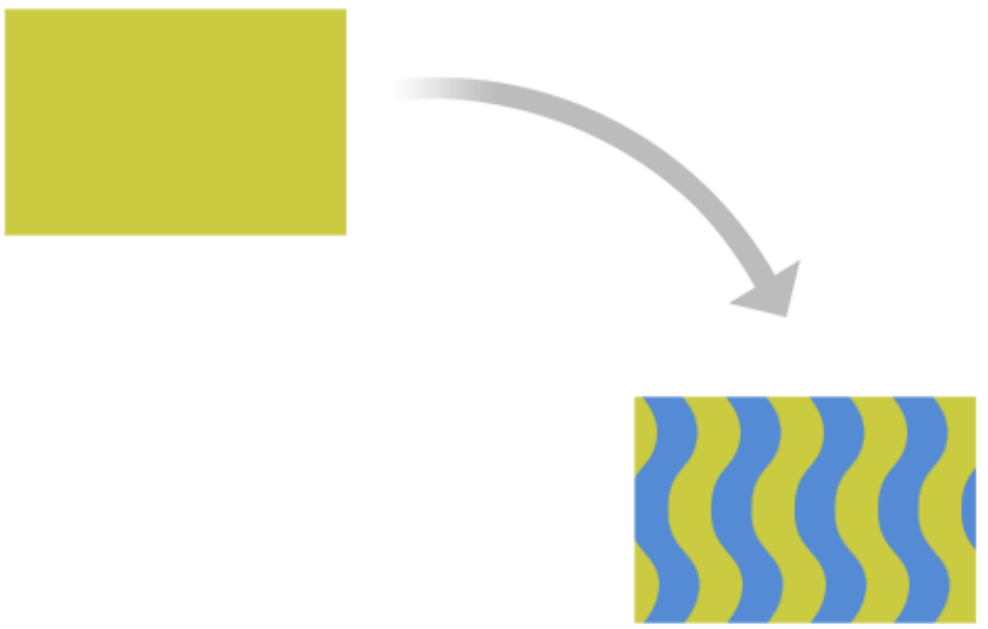


このスライドでは、FOSSを使う際の「改変」の意味について概説しています。

This slides outlines what modification means when using FOSS.

翻訳する（Translation）

開発者は、コードをある状態から異なる状態に変換することができる。



例として以下のようなものがある：

- 中国語から英語への翻訳
- C++ からJavaへの変換
- バイナリへのコンパイル

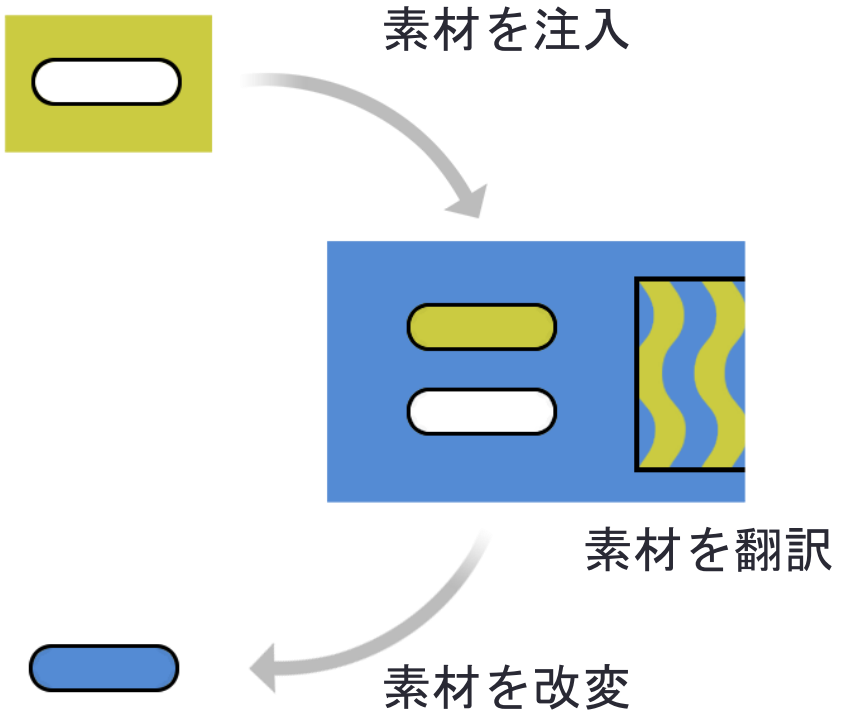
このスライドでは、FOSSを使う際の「翻訳」の意味について概説しています。

This slides outlines what translation means when using FOSS.

開発ツール

開発ツールがこれらの操作のいくつかをバックグラウンドで実行してくれる場合がある。

たとえば、開発ツールのコード部分を出力ファイルに挿入してくれるものがある。



このスライドでは、開発ツールが「裏方となって」これらのアクションを実施する場合があることを説明しています。この内容は企業によく知っておいていただきたいところです。

This slides explains that development tools may do some of these actions “behind the scene”, and this is an area that companies should be aware of.

FOSSコンポーネントをどのように頒布するか？

- 誰がソフトウェアを受け取るのか？
  - 顧客／パートナー
  - コミュニティ プロジェクト
  - 企業集団内にある別法人※（頒布として扱う場合がある）

- 頒布用のフォーマットは何か？
  - ソースコードでの頒布
  - バイナリでの頒布
  - ハードウェアにプレインストール

※たとえばグループ内の会社間（親会社から子会社、その逆など）での提供

このスライドでは、頒布することの背景にあるいくつかの考え方を説明しています。これはFOSSライセンスは通常、頒布の期間内に適用されるものであるためです。この点はコンプライアンスプログラムで考慮すべき重要なポイントです。

This slide explains some of the concepts behind distribution. Because FOSS licenses usually apply during distribution, this is a key point to consider in a compliance program.

理解度チェック

OPENCHAIN

- 取り込むとはどういうことですか？
- リンクするとはどういうことですか？
- 改変するとはどういうことですか？
- 翻訳するとはどういうことですか？
- 頒布を検討する上で重要な要素は何ですか？

取り込みとはFOSSコンポーネントの一部を自身のソフトウェア製品にコピーすることです。

リンクとは自身のソフトウェア製品とFOSSコンポーネントをリンク（Link）もしくは接合（Join）することです。

改変とはFOSSコンポーネントに変更を加えることです。

翻訳とはコードをある状態から別の状態に変換することです。

オープンソースを頒布することを考える際には、以下の2つのことを考える必要があります。
そのソフトウェアを受け取るのは誰か？
- 顧客／パートナー
- コミュニティ プロジェクト
- 企業集団内にある別法人（頒布として扱う場合がある）

頒布フォーマットは何か？
- ソースコードによる頒布
- バイナリによる頒布
- ハードウェアにプレインストール

Incorporation is when you copy portions of a FOSS component into your software product.

Linking is when you link or join a FOSS component with your software product.

Modification is when you make changes to a FOSS component.

Translation is when you transform the code from one state to another.

When thinking about distribution of Open Source you should consider two things:
Who receives the software?

- Customer/Partner
- Community project
- Another legal entity within the business group (this may count as distribution)

What is the format for delivery?

- Source code delivery
- Binary delivery
- Pre-loaded onto hardware

第5章

FOSSレビューの実施

FOSSレビュー

- プログラムマネージャー、プロダクトマネージャーおよびエンジニアに提案されたFOSSコンポーネントの有益性や品質面のレビューを実施しその後、選択されたコンポーネントの使用に付随する権利や義務についてのレビューが開始される
- FOSSコンプライアンス プログラムにとって鍵となる要素が「FOSS レビュー」のプロセスであり、これにより企業は使用するFOSSを分析し、権利と義務を理解することができる
- FOSSレビューのプロセスには以下のステップがある：
  - 関連情報の収集
  - ライセンスの義務の分析と理解
  - 企業のポリシーや事業目標に合わせた指導の提供

FOSSレビューはFOSSコンプライアンス プログラムの基本的構成要素です。

FOSSレビューはエンジニアリング チーム、ビジネス チーム、および法務チームが集まる場となりえます。より大規模に首尾よく行うために、計画や組織化を必要とする場合があります。

- 関連情報収集においてエンジニアリング チームもしくは開発チームが参加することもあります。
- 法務チームはライセンスの義務について分析、決定を下し、指導を行います。
- ビジネスおよびエンジニアリング チームは指導を受けて、実装します。

The FOSS Review is a basic building block of a FOSS Compliance Program.

A FOSS Review can be the meeting point for engineering, business and legal teams, and can require planning and organization to successfully conduct on a large scale.

- Engineering or developer teams may participate in gathering relevant information
- Legal teams analyze and determine license obligations and provide guidance
- Business and engineering teams may receive and implement guidance

FOSSレビューの開始

OPENCHAIN

プログラム マネージャー、プロダクト マネージャー、エンジニア、法務関係者など、企業内でFOSSに関わる全員がFOSSレビューを開始することができる。

注：このプロセスは、エンジニアリング部門もしくは外部ベンダーによって新たなFOSSベースのソフトウェアが選定された時に開始されることが多い

最初のステップはFOSSレビューを開始するために適切な参加者を特定することです。

以下のような問いかけが重要です：

- FOSSの使用について誰が意思決定者なのか（マネージャー、アーキテクト、個々の技術者など）？
- FOSSの使用について彼らはどのように質問・疑問を上げることができるのか？
- 開発プロセスの中にFOSSレビューが開始できる定まったチェックポイントがあるか？

The first step is to identify the proper parties to initiate a FOSS Review

Important questions to ask include:

- Who are the decision makers about FOSS usage (managers, architects, individual engineers, etc.)?
- How can they raise questions about FOSS usage?
- Is there a regular point in your development process where FOSS Reviews can begin?

どのような情報を集める必要があるか？

OPENCHAIN

FOSSの使用分析にあたり、FOSSコンポーネントの属性、起源、使用方法などの情報を集める。たとえば以下のようなものがある。

- パッケージ名
- パッケージを取り巻くコミュニティの状況（活動状況、メンバーの多様性、反応の早さ）
- 版名（バージョン）
- ダウンロード元もしくはソースコードのURL
- 著作権保有者
- ライセンスおよびライセンスのURL
- 帰属表示やその他の告知/通知/表示、それらのURL
- 意図して行われた改変に関する記述
- 依存関係のリスト
- 製品で意図している使用方法
- そのパッケージを内包する製品のファースト リリース（最初の公開・販売）
- ソースコードがメンテナンスされているロケーション
- 過去に別経緯でそのパッケージに対して下された承認の可能性
- 外部ベンダーからの提供物の場合：
  - 開発チームのコンタクト ポイント
  - 著作権表示、帰属表示、およびライセンスの義務履行に必要なベンダー改変ソースコード

注目すべきは、この情報のリストが 非常に多く見えることです。しかし、必要とされる情報量はFOSSコードを取り扱おうとする企業の規模、および、FOSSをどのように取り扱うかに依存します。大規模な組織体は小規模なものよりも多くの情報を必要とする傾向があります。

外部ベンダーを利用した場合は、いくつか付加的な論点があります。まず、将来的にFOSSに関する問題が生じた場合、そのベンダーを追跡調査する必要があるかもしれません。その際に信頼できるコンタクト先が重要となります。またベンダーから引き渡されたFOSSに対しライセンスの義務を果たす必要があるかもしれません。そういった義務を果たすべく必要性に応じて告知／表示やソースコードがあることを確かめましょう。

It should be noted that this list of information looks quite large. However, the amount of information required depends on the size of your company and what you intend to do with the FOSS code. Large entities tend to require more information than small entities.

There are a couple additional issues in the case of external vendors. First, you may need to follow up with the vendor if FOSS issues arise in the future, and having a reliable point of contact is important. You may also need to meet FOSS license obligations for FOSS delivered from the vendor. Ensure you have the notices and source code as needed to meet these obligations.

FOSSレビューチーム

FOSSレビュー チームには企業でFOSSの使用を支援、指導し、とりまとめ、レビューする代表者たちが含まれる。その代表者には、以下が含まれる場合がある：

- ライセンスの義務を特定し、評価する法務関係者
- FOSSの使用の確認と追跡を支援するソースコードスキャン実施やツールサポートを提供する関係者
- 事業企画、商用ライセンス、輸出コンプライアンスなどを取り扱い、FOSSの使用によって影響を受ける可能性のある各部門の専門家（技術的観点）

FOSSレビュー チームは異なった分野にまたがって構成されます。

法務チームは、社内もしくは外部の弁護士を含めることができ、ライセンスの義務に応じたFOSS使用をレビューし、評価します。

法務チームは、次のように他のチームからサポートされる場合もあります。

- FOSSの使用を特定し、追跡する調査・分析チーム。このチームはコードベース（ソースコードの集積場所）に存在するFOSSコンポーネントを特定するためのコードスキャンツールやフォレンジクス（法的確証収集）ツールを駆使した支援などを行います。また本チームは、後続コンプライアンスプロセスを支援するべく、FOSSの使用について収集した情報を整理し、追跡することも実施します。
- その他に、商用ライセンスや輸出コンプライアンスおよび事業企画チームなど、FOSSに関連する論点で影響を受けうる専門家、代表者も想定されます。

The FOSS Review team may consist of an interdisciplinary team

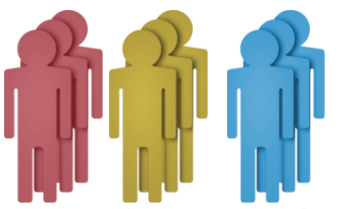
The legal team, which may include in-house or outside attorneys, reviews and evaluates the FOSS usage for license obligations

The legal team may be supported by others, including:

- Scanning and tooling teams that identify and track FOSS usage. These teams may provide support using code scanning or forensics tools to identify FOSS components in a codebase. The teams may also organize and track information gathered regarding FOSS usage to assist with later compliance processes.
- Other specialists or representatives that may be impacted by FOSS-related issues, such as commercial licensing, compliance or business planning teams.

提案されたFOSSの使用を分析する

OPENCHAIN



FOSSレビューチームは指導を行う前に、たとえば以下のような論点に対し、収集した情報を査定する必要がある。情報の正確さを確認するためのコードスキャンの実施がこれに含まれることがある

FOSSレビューチームは以下を考慮する必要がある：

- コードと付随した情報が、完全で、一貫していて、正確か？
- 宣言されたライセンスがコードファイルにある内容と合致しているか？
- ソフトウェアを構成する他のコンポーネントとともに使用することをライセンスが許容しているか？

FOSSレビューチームは、FOSSの使用を適切に評価するための専門知識を有する必要があります。FOSSの使用案について法務チームやビジネスチームを教育するためにエンジニアリング チームの支援が必要となることもあります。たとえば、FOSSの明らかになっていない使用を見つけるためにコード スキャンツールが使われることがあります。

FOSSの使用案が十分査定されると法務チームは判断を下す際に必要な情報を得たことになります。

The FOSS Review team should have the expertise to properly assess the FOSS usage. This may require support from engineering teams to educate legal and business teams about the proposed FOSS usage. For example, code scanning may be used to locate undisclosed FOSS usage.

Once the proposed FOSS usage has been fully assessed, the legal team will then have the necessary information on which to make its judgments.

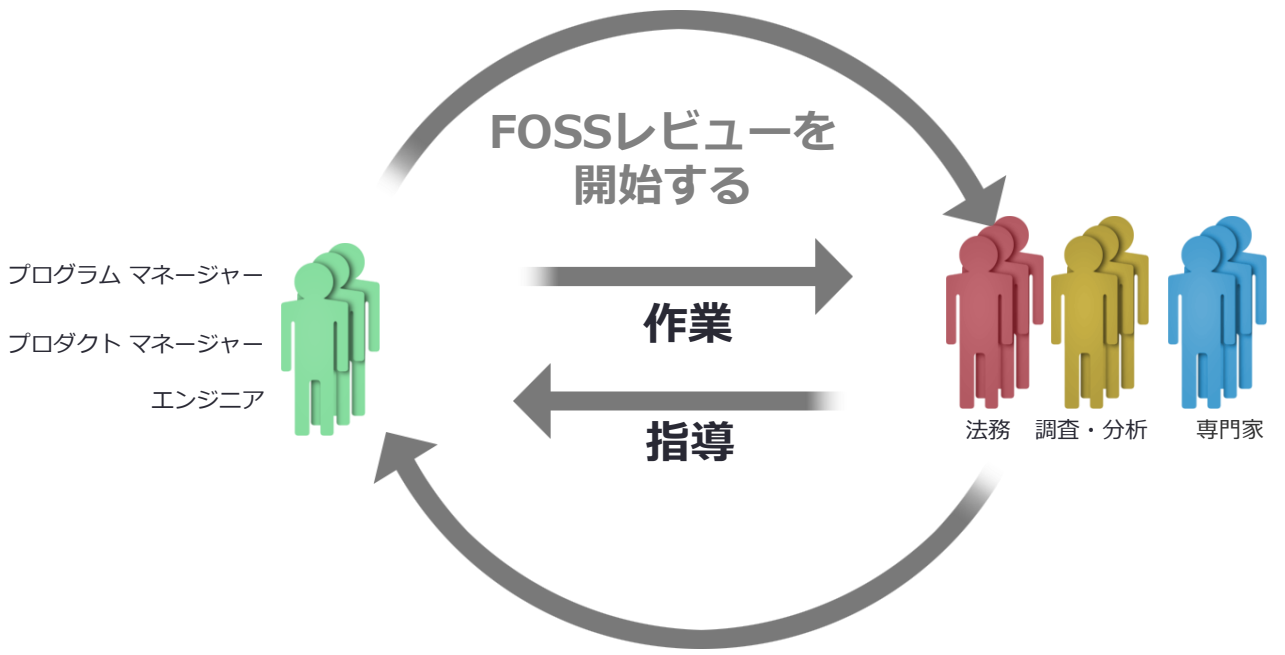
ソースコード スキャン ツール

- ソースコードスキャンを自動化するツールは数多く、さまざまなものが存在
- それらのすべては特定のニーズに向けたソリューションであるため、あらゆる課題を解決するものではない
- 企業はそれらの中で自分たちの特定の市場領域や製品に合うものを選定する
- 多くの企業は自動化ツールと人手によるレビューを併用している
- 無償で、自由に利用できるソースコード スキャン ツールの1つのよい例としてThe Linux Foundation でホストしたプロジェクトである、FOSSologyがある：https://www.fossology.org

本スライドではオープンソースコードスキャンツールがどんなもので、それがどういった働きをし、経験の浅いユーザがこのトピックについてどのように知識を集めることができるのか、といった点について全体像で説明しています。

This slide explains the big picture of what Open Source code scanning tools are, how they work, and where a new user can start to gather knowledge about the subject.

FOSSレビューの遂行



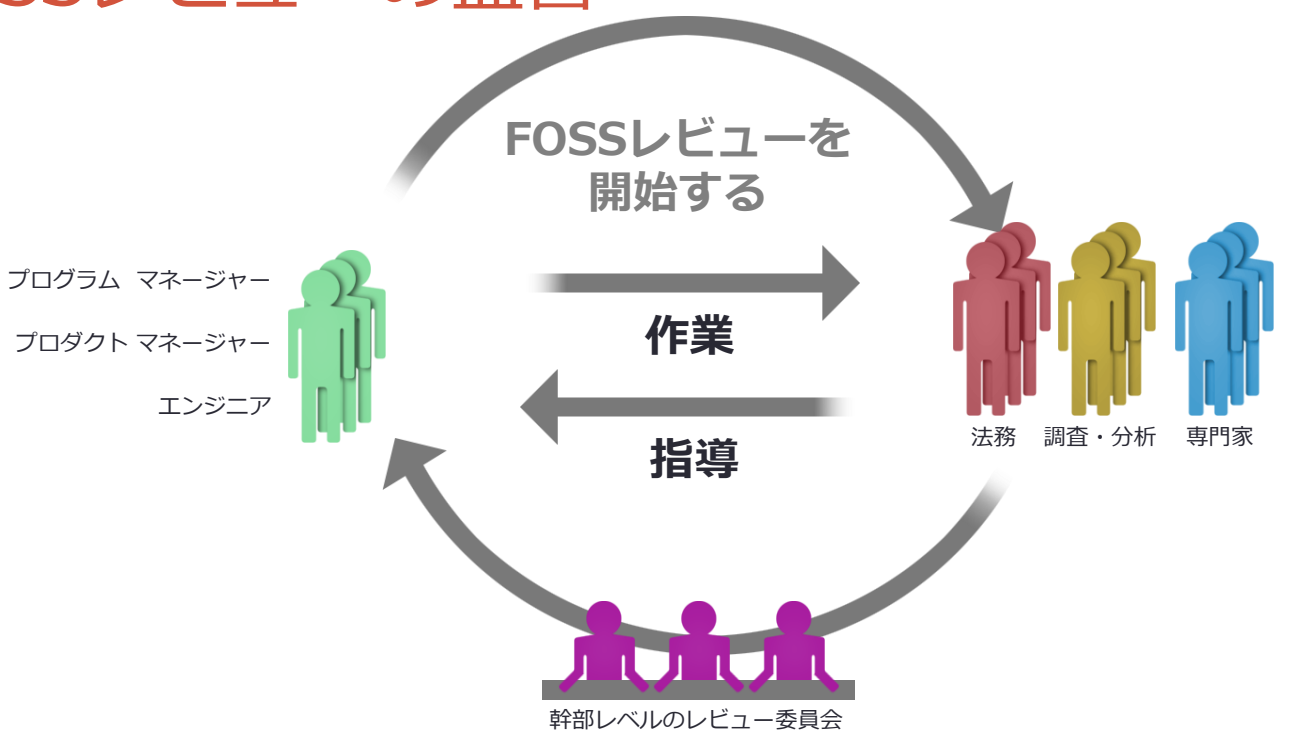
FOSSレビュープロセスは、エンジニアリング チーム、ビジネス チーム、法務チームなど分野をまたぐ形となる。これらのグループが正確に問題を理解することを確かなものとするにはインタラクティブな取り組みであることが必要となる。また、明確で、皆に共有される指針をここで作り出すことができる。

FOSSレビューのプロセスは、利害関係のある参加者が協力できるように、柔軟なものである必要があります。時としてFOSSの使用シナリオがFOSSレビュー チームにとって明確でないこともあります。エンジニアリング チームは、より深くインプットを提供するための技量が必要となるでしょう。同様に、エンジニアリング チームは、FOSSレビュー チームからの指導を実行に移す際に支援を必要とするかもしれません。

--

The FOSS Review process should be flexible enough to allow the interested parties to collaborate. Sometimes a FOSS usage scenario may not be clear to the FOSS review team. The engineering team will need the ability to provide further input. Likewise, the engineering team may need assistance in implementing guidance from the FOSS review team.

FOSSレビューの監督



FOSSレビューのプロセスにおいては、関係者間での意見不一致の解決や最重要となる意思決定を承認するべく、幹部レベルでの監督機能が必要となる

FOSSレビュープロセスは監督機能を持つ必要があります（たとえば、この図では幹部レベルのレビュー委員会）。このような監督委員会は、重要な方針決定や、レビュープロセスでの関係者における意見不一致の解決などを行います。

The FOSS Review process should have oversight (for example, an Executive Review Committee in this diagram). The oversight committee may make important policy decisions or resolve disagreements between parties in the review process.

理解度チェック

OPENCHAIN

- FOSSレビューの目的は何ですか？
- FOSSコンポーネントを使いたい時に最初に行うべきアクションは何ですか？
- FOSSの使用に関する質問や疑問がある場合、何をするべきですか？
- FOSSレビューのためにどのような種類の情報を集めますか？
- 誰がそのソフトウェアのライセンスを供与しているのかを確認するには、どのような情報が役立ちますか？
- 外部ベンダーから受領したコンポーネントをレビューする際に追加的な情報として重要なものは何ですか？
- FOSSレビューで収集された情報の質を評価するためにどのようなステップをとることができますか？

FOSSの使用に関する情報を収集し、分析するため、および適切な指導を行うためです。

FOSSレビュープロセスを開始します。このプロセスを開始する手法は企業によって異なりますが、開発でOSSの使用に関わる人たちにはオープンにするべきです。

FOSSレビュー プロセスを開始するか、FOSSレビュー チームにコンタクトをとります。組織においてFOSSの使用者が指導を受け入れることができるよう、そのプロセスは柔軟であるべきです。

第一歩としては、パッケージ名、版名（バージョン番号）、ダウンロード元URL、ライセンス、説明、製品内で意図される使用法などがあるとよいでしょう。組織や意図したユースケースに依存して正確かつ詳細な情報が必要となるでしょう。

通常、著作権表示、帰属情報およびソースコードによって誰がそのFOSSソフトウェアをライセンスしているかを特定することができます。

将来発生しうるFOSSの問題を追跡するために必要な開発チームのコンタクト ポイントです。外部ベンダーのソフトウェアをコントロールするFOSSライセンスの義務を履行するために、著作権表示、帰属表示、およびベンダーの改変に対応したソースコードを入手する必要があるかもしれません。

完全性、一貫性、正確性について情報をチェックすることです。このプロセスは、開示されていないFOSS使用に対してコード スキャン ツールで精査することも含めて支援チームの助けを借りることができます。

To gather and analyze information regarding FOSS usage and to produce appropriate guidance.

Initiate a FOSS review process. The method for initiating this process may vary by company, but should be open to those who are involved in using FOSS in development.

Initiate a FOSS review process or contact the FOSS review team. The process should be flexible enough so that FOSS users in your organization have access to guidance.

The package name, version, download URL, license, description and intended use in your product is a good starting point. The precisely level of detail you will need depends on your organization and intended use case.

The copyright notices, attribution and source code normally helps to identify who is licensing the FOSS software.

Development team's point of contact in case you need to follow up with future FOSS issues. You may also want to obtain copyright and attribution notices, and source code for vendor modifications if these are needed to satisfy license obligations for FOSS licenses governing the third party software.

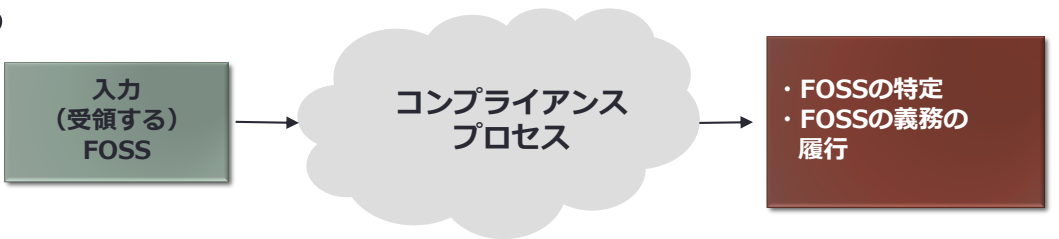
Check information for completeness, consistency and accuracy. This process may be assisted by support teams, including teams that run code scanning tools to scan for undisclosed FOSS usage.

第6章

コンプライアンス マネジメントの始めから終わりまで（プロセス例）

概要

- コンプライアンス マネジメントは、製品の中で使われるFOSSコンポーネントを管理する一連のアクション。企業の商用コンポーネントについても同様のプロセスが実施されている場合がある。FOSSコンポーネントはOpenChain 仕様書で「供給ソフトウェア（Supplied Software）」と呼ばれる
- アクションとしては以下が含まれることが多い
  - 供給ソフトウェアにおけるすべてのFOSSコンポーネントの特定
  - それらのコンポーネントによって生まれるすべての義務の特定と追跡
  - すべての義務が履行され、将来にわたり履行されることの確認
- 小規模の企業ではシンプルなチェック リストを、大企業においては詳細なプロセスを用いる



このスライドは、コンプライアンスマネジメントの定義と最終目標について述べています。

本章は大企業で実施される可能性のある具体的な例を提供します。小規模な企業では、より簡素化したプロセスで取り組むことが望まれるでしょう。

This slide describes the definition of compliance management and its end goals.

Note that this section provides a detailed example of what may take place in a large enterprise. Smaller companies may wish to approach the process in a more streamlined way.

中・小規模企業のチェックリストの例

OPENCHAIN

継続的に行うコンプライアンス関連タスク：

1. すべてのFOSSを調達／開発サイクルの早期で発見する
2. すべての使用FOSSコンポーネントをレビューし、承認する
3. FOSSの義務を満たすために必要となる情報を検証する
4. 社外FOSSプロジェクトへのコントリビューションをレビューし承認する

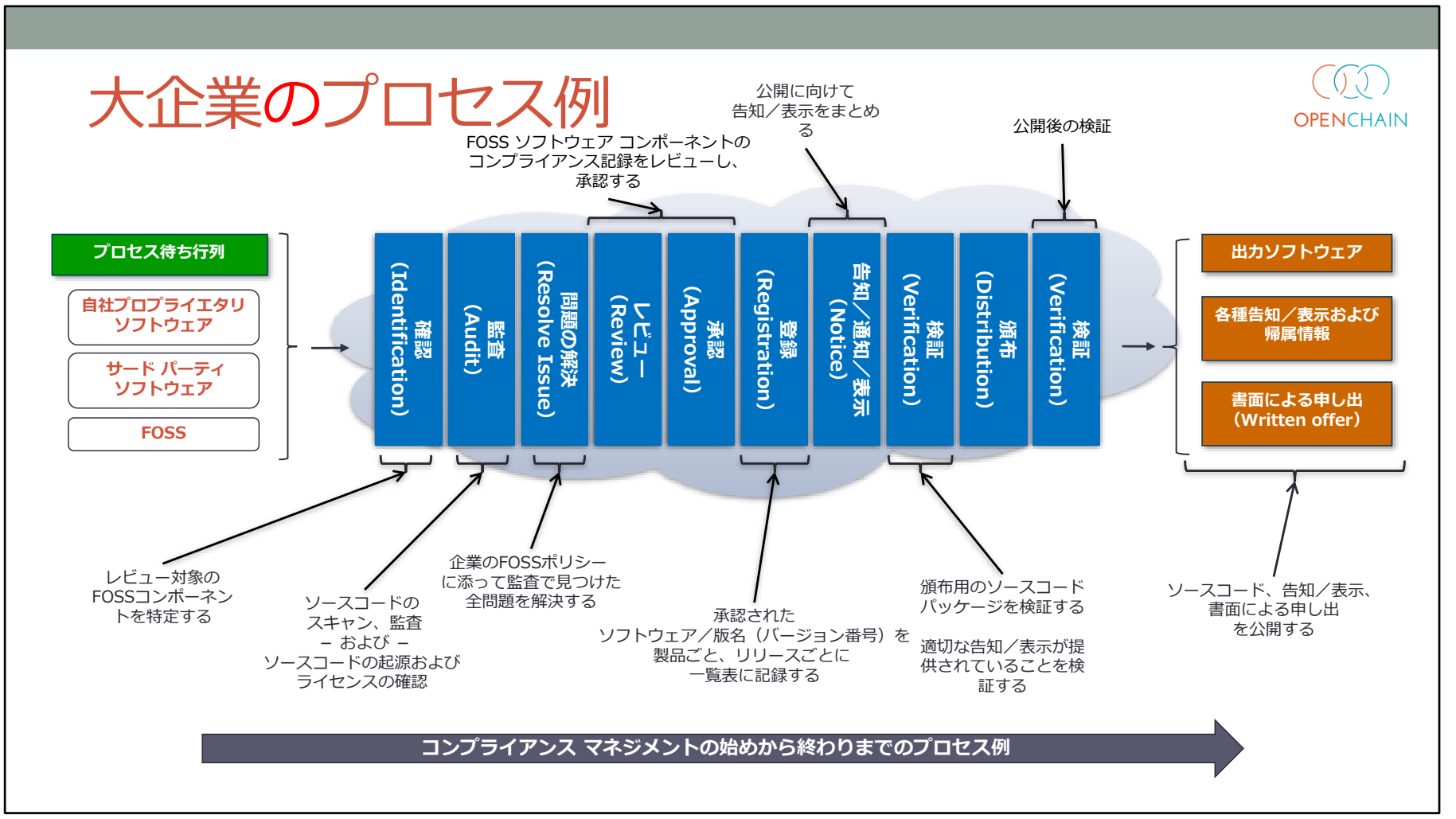
支援するための要求事項：

1. 適切なコンプライアンス対応要員を配置し、明確な責任ラインを指定する
2. 既存のビジネスプロセスをFOSSコンプライアンスプログラム支援のために適合させる
3. 組織のFOSSポリシーについて全社員が学習できるトレーニングを持つ
4. FOSSコンプライアンスに関係するすべての活動の進捗を追跡する

これらの項目についての詳細チェックリストはこちらで入手可能： https://www.linuxfoundation.org/projects/opencompliance/self-assessment-compliance-checklist

本スライドは、中・小規模企業（SME）が有効なコンプライアンス プログラムを構築し、展開するために実施必要となりうる事項について説明しています。

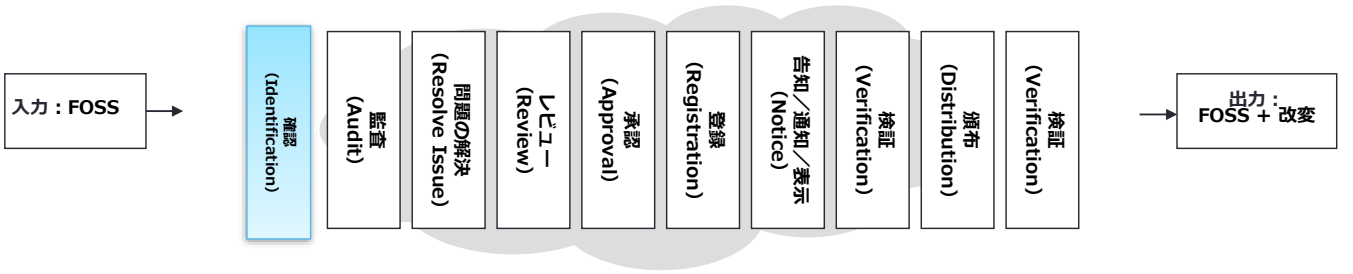
This slide describes what a Small to Medium Enterprise (SME)might need to do to build and deploy an effective compliance program.



本スライドは、大企業のプロセスとして使用されることを想定した、各ステップの全体像です。

This slide is an overview of the steps that a larger enterprise might use for their process.

FOSSの使用を確認し、追跡する



すべてのソースに含まれるFOSSを確認し、追跡を開始する

- ステップ：
  - エンジニアリング部門からリクエストが入力される
  - 当該ソフトウェアのスキャンを実施する
  - サードパーティのソフトウェアに対する精査を実施する
  - ソース リポジトリに追加された、新たなコンポーネントを人手で認識する
- 成果：
  - そのFOSSについてコンプライアンスの記録が作成（またはアップデート）される
  - FOSSポリシーの規定に則り、網羅的もしくは限定的と定めた範囲でソースコードレビューのための監査（ 次のステップ） が要請される

この例における最初のステップは、FOSSの使用を確認することです。

たとえば開発チームがリクエストを上げた（またはFOSSレビューを開始した）場合などにステップが開始されます。またこのステップは、出荷ソフトウェアにFOSSが使用されている、または企業が使用するサード パーティ ソフトウェアにFOSSが使用されていること、そしてそのために適正なレビューの実施が必要であることをレビュー チームが発見した場合や、通知された場合にも開始します。

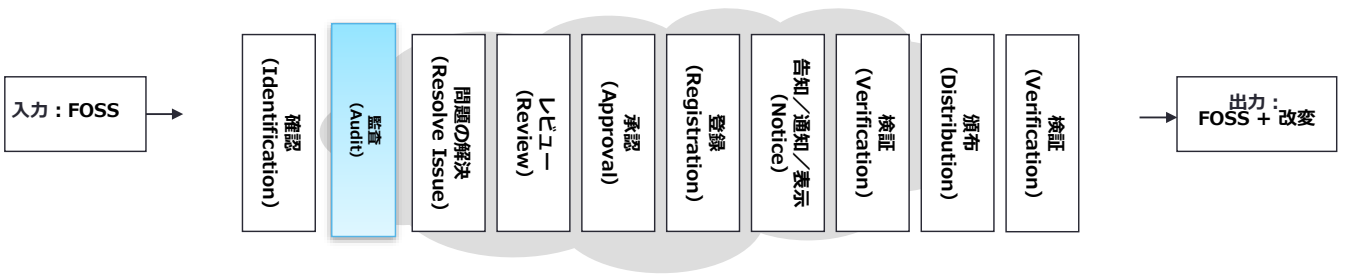
この例では、FOSSレビュー チームはエンジニアたちからのレビュー リクエストを通じて、内部開発・サードパーティのソフトウェアへスキャンを実施することによって、あるいは、開発のブランチにチェックインされたコードのレビューによってFOSSの使用を確認します。その後レビュー チームはレビュー記録を生成し、次の「監査」ステップに進みます。

The first step in our example process is to identify FOSS usage.

For example, a development team may have initiated a request (or initiated a FOSS Review). The step may also begin if the review team discovers or is notified that FOSS is being used in a software release or in third party software used by the company, and that a proper review needs to take place.

In this example, the FOSS review team may identify FOSS usage through review requests from engineers, from performing scans of internally-developed and third-party software, or reviewing code checked into development branches. The review team will then create a record of the review, then move to the next step (“Audit”).

ソースコードを監査する



FOSSコンポーネント、およびその起源とライセンスを確認する

- ステップ：
  - 監査のためのソースコードが特定される
  - ソフトウェア ツールによってソースがスキャンされる
  - 監査やスキャンによって「ヒット」したものがレビューされ、コードの起源が適正かどうかが検証される
  - ソフトウェアの開発／リリースの ライフサイクルをベースに監査もしくはスキャンが繰り返し実施される
- 成果：
  - 以下を特定した監査レポート：
  1. ソースコードの起点とライセンス
  2. 解決が必要な問題

次のステップは、前のステップで確認されたソースコードの監査です。

ここでの例では、企業は確認されたFOSSコンポーネントについて調査を実施しています（たとえば、宣言されているライセンスのレビューや、FOSSコンポーネントの起源の調査など）。また企業はソースコードの起源や構成を検証するためにスキャンも実施します。

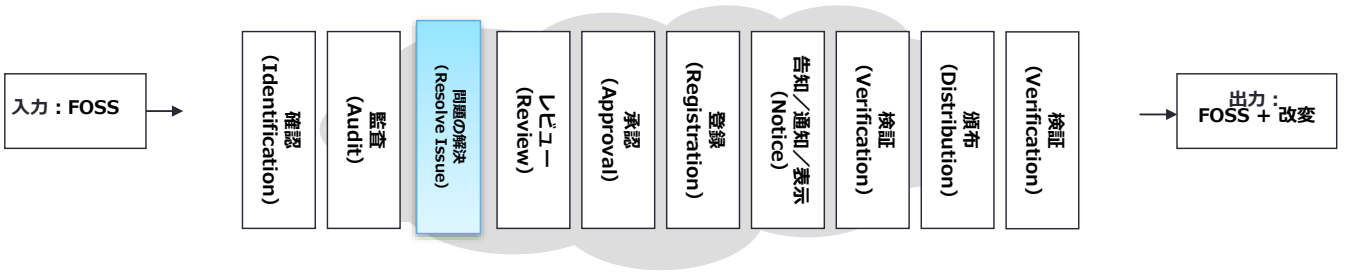
その後レビュー チームは、ソースコードの起源とライセンスに関して結論づけた監査レポートを作成します。

The next step is auditing source code identified in the previous step.

In our example, the company may conduct research into the identified FOSS component (e.g., review declared licenses, research origins of the FOSS component). The company may also scan the source code to verify the origin and composition of the code.

The review team may then produce an audit report with its conclusions regarding the origin and licensing of the source code.

問題を解決する



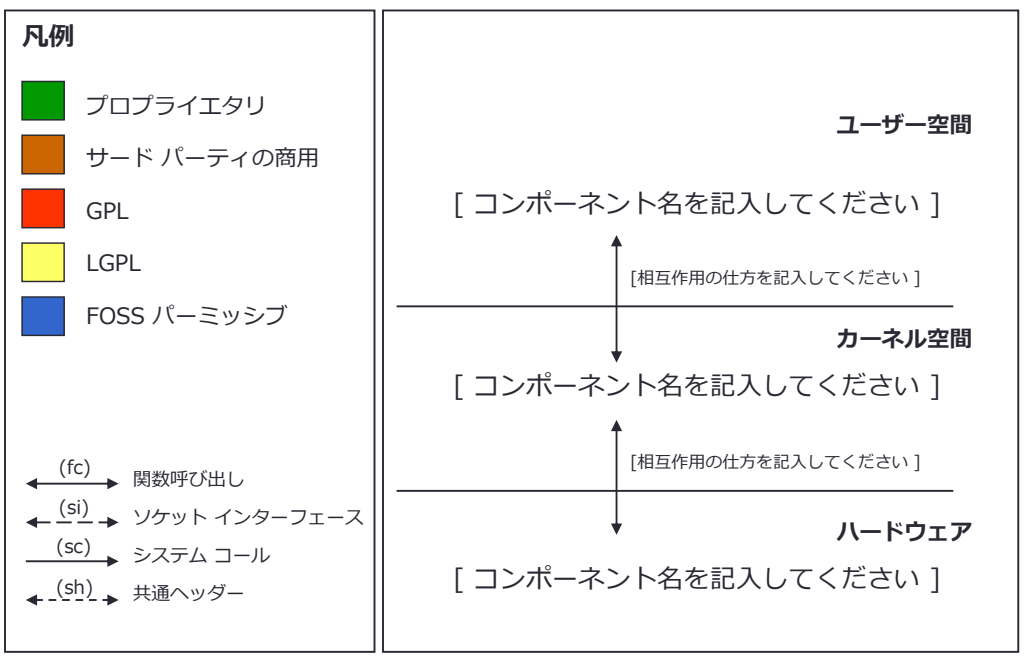
監査で確認されたすべての問題を解決する

- ステップ：
  - 監査レポートで指摘されたFOSSポリシーに反する問題を解決するために、適切なエンジニアにフィードバックを提供する
  - その後、適切なエンジニアが関連するソースコードに対しFOSSレビューを実施する（次スライド参照）
- 成果：
  - レポートでフラグを立てられたそれぞれのファイルに対する問題の解消、およびフラグの立てられたすべてのライセンスに関する矛盾の解決

ソースコードの起源とライセンスを確認した監査レポートが作成されると、レビュー チームは企業のFOSSポリシーに従い、すべての問題にフラグを付け、レビューをする必要があります。たとえば、以前のステップで両立しないライセンス下にある異なるFOSSのコードを含んだFOSSコンポーネントを特定したとします。レビューチームはこの問題を解決するためにエンジニアリングチームに適切なフィードバックを提供する必要があります。

Once an audit report is produced that confirms the origin and licensing of source code, the review team should flag and review any issues under the company FOSS policy. For example, the earlier steps may have identified a FOSS component that contains other FOSS code under an incompatible license. The review team should provide appropriate feedback to the engineering team to resolve the issues.

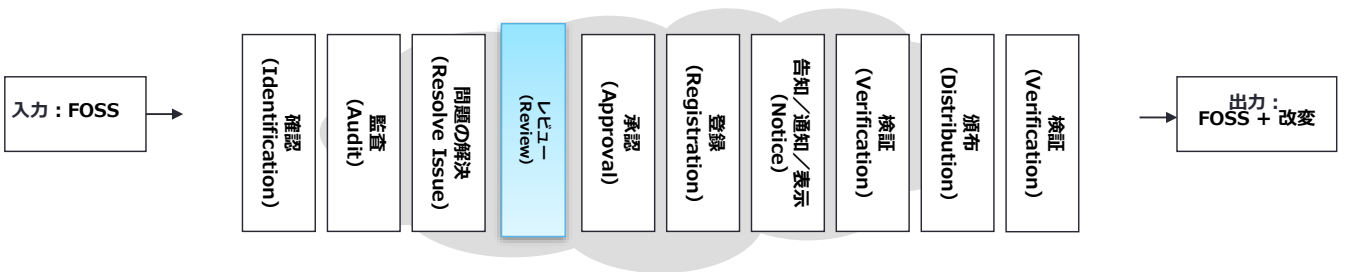
アーキテクチャ レビュー（テンプレートの例）



このスライドでは、FOSSの使用と企業のソフトウェアとの関係を説明するために使うテンプレートを掲載しています。たとえば、FOSSと企業のコンポーネントはー緒にリンクされるのか？といったことです。このようなテンプレートは、計画されたFOSSの使用についてFOSSレビューチームの理解を助けるためにエンジニアリング チームによって作成されることもあります。

This slide contains a template that may be used to illustrate FOSS usage and its relationship with company software. For example, how are FOSS and company components linked together? Templates such as these may be created by engineering teams to help educate the FOSS review team about planned FOSS usage.

レビューを実施する



監査レポートをレビューし、発見されたすべての問題が解決していることを確認する

- ステップ：
  - レビュー スタッフに適切な職権レベルを含める
  - FOSSポリシーに則ったレビューを実施する
- 成果：
  - 監査レポートにあるソフトウェアがFOSSポリシーと合致することを確かなものとする
  - 監査レポートで発見されたことを保存し、解決された問題を次のステップへの準備ができた（つまり承認された） ものとして示される

このステップでは、FOSSレビュー チームが直前のステップで収集された事実をレビューし、FOSSライセンス下で企業が負うべき義務を確認します。

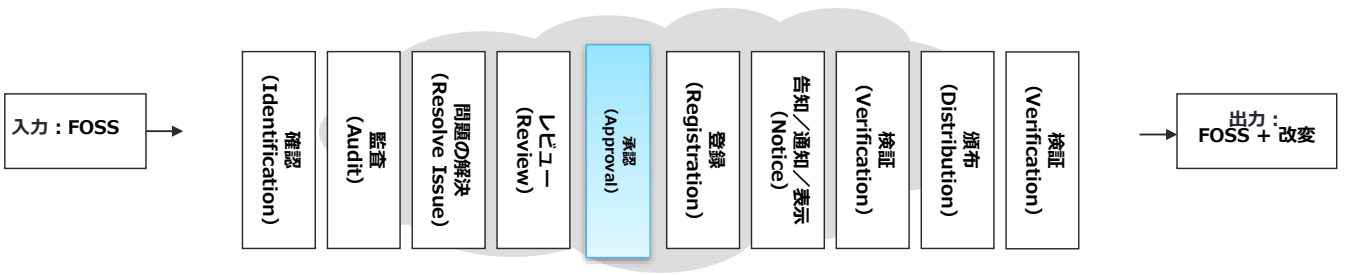
このステップは直前のステップ（監査での問題を解決する）と密接に関係しています。直前のステップでは企業のポリシーと合致しないFOSSの使用を取り除きました。このステップでは使用していくことになったFOSSのライセンス義務を評価し、確認します。

In this step, the FOSS review team reviews the facts collected in the previous steps and identifies the company’s obligations under the FOSS licenses.

This step may be closely linked with the previous step (Resolving Audit Issues). In the previous step we removed FOSS usage that did not conform to company policy. In this step, we evaluate and identify the license obligations for FOSS usage that is retained.

承認

- 前ステップのソースコード監査およびレビューの結果に基づき、ソフトウェアの使用が承認、却下される
- この承認で、承認対象のFOSSコンポーネントのバージョン、使用方法、およびFOSSライセンス下で適用されるその他すべての義務などを明確にする
- 承認は適切な職権レベルで行われる必要がある

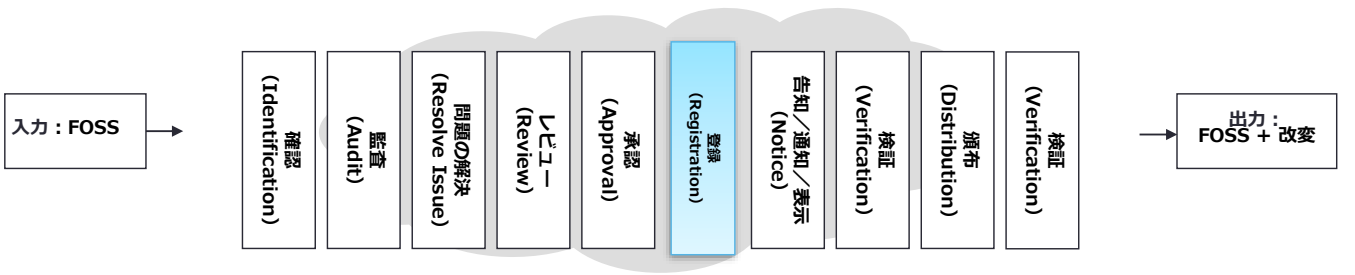


ここでの例における承認ステップでは、レビュー チームは問題のFOSSの使用を、それに伴う条件や義務に添って承認するかどうかを明らかにします。この承認では、FOSSコンポーネントの版名や承認される使用シナリオなどの重要な詳細情報を盛り込む必要があります。

In the approval step of our example process, the review team communicates whether it approves of the FOSS usage in question, along with any associated conditions or obligations. The approval should also include important details such as version numbers of FOSS components and the approved usage scenario.

登録／承認の追跡

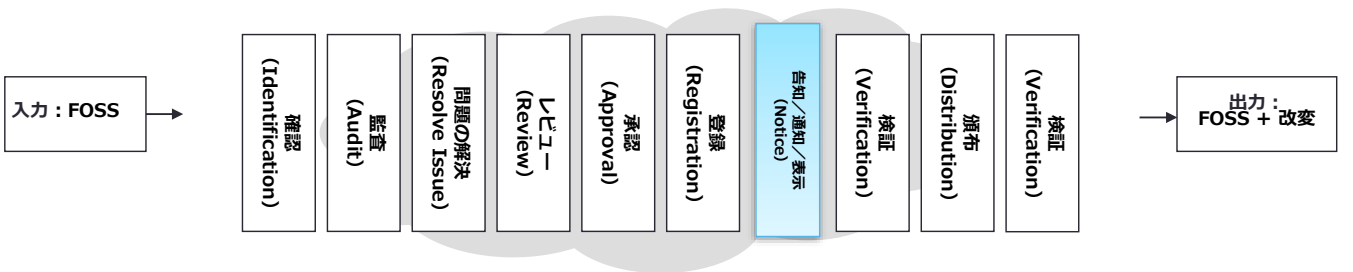
- 製品内での使用についてFOSSコンポーネントが承認された場合、それがその製品のソフトウェア一覧表に追加される
- 承認内容とその条件が追跡システムに登録される
- 新しいバージョンのFOSSコンポーネントや新しい使用方法が提案された場合には、新たな承認が必要となることを追跡システムで明確にする



前ステップの承認情報は、そのソフトウェアのリリースに関与するすべての人々が理解し、関連するライセンスの義務を履行できるように、登録され、追跡される必要があります。

Approval information from the previous step should be tracked or registered so that anyone releasing the software can understand and comply with the relevant license obligations.

告知／通知／表示



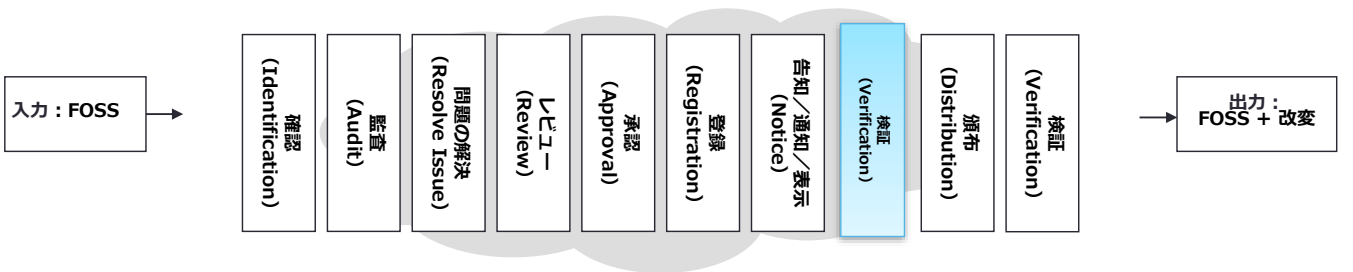
製品リリース時に用いる適切な告知／表示を準備する

- 著作権表示と帰属表示のすべてを提供することで、FOSSが使用されていることを表明する
- 製品のエンドユーザーにFOSSソースコードの写しの入手方法に関する情報を提供する（GPLやLGPLのケースのように、その必要がある場合）
- 必要に応じ製品に含まれるFOSSについてライセンス同意書全文のコピーを用意する

FOSSライセンスで求められる場合、適切な告知／表示文が準備されなければなりません（多くの場合、製品に添付されるテキストファイルで）。告知／表示には帰属表示や改変告知、あるいは、ソースコード提供の申し出が含まれます。いくつかのライセンスについては、ライセンス全文の写しを含める必要があります。

If required by a FOSS license, appropriate notices should be prepared (often in a text file that accompanies the release). Notices may include attribution notices, modification notices, or offers for source code. For some licenses, you may also need to include a full copy of the license text.

頒布前の検証



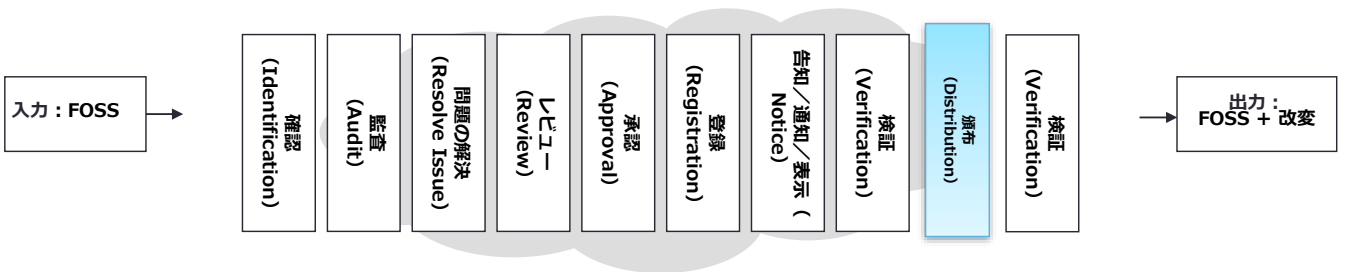
頒布されるソフトウェアがレビューされ承認されたことを検証する

- ステップ：
  - 頒布用のFOSSパッケージが明確になっていて、承認されていることを検証する
  - レビューされたソースコードが製品として出荷されるバイナリ形態の同等物と合致していることを検証する
  - エンドユーザー向けに当該FOSSのソースコードをリクエストできる権利について情報提供するための適切な告知文がすべて用意されていることを検証する
  - 確認されたその他義務の履行を検証する
- 成果：
  - 頒布パッケージには、レビューされ承認されたソフトウェアだけが含まれている
  - （OpenChain仕様書で定義される）「頒布コンプライアンス関連資料」として、頒布パッケージやその他頒布形態に適切な告知／表示が盛り込まれている

例として挙げたここでのプロセスについて、このスライドでは企業がリリース前にFOSSライセンスの義務を履行したことを検証していきます。ソースコードを入手可能としなければならない場合、企業はソースコードが頒布されるバイナリ ファイルと合致していることを検証します。また企業は告知文が適切に生成され、頒布パッケージに含まれていることを必要に応じて検証します。

In this slide of our example process, the company verifies that it has met its FOSS license obligations before release. In cases where source code must be made available, the company verifies that the source code matches the binary files being distributed. The company also verifies that notices are properly produced and included in distribution packages as needed.

添付ソースコード※を頒布する



添付ソースコードを要求された形で提供する

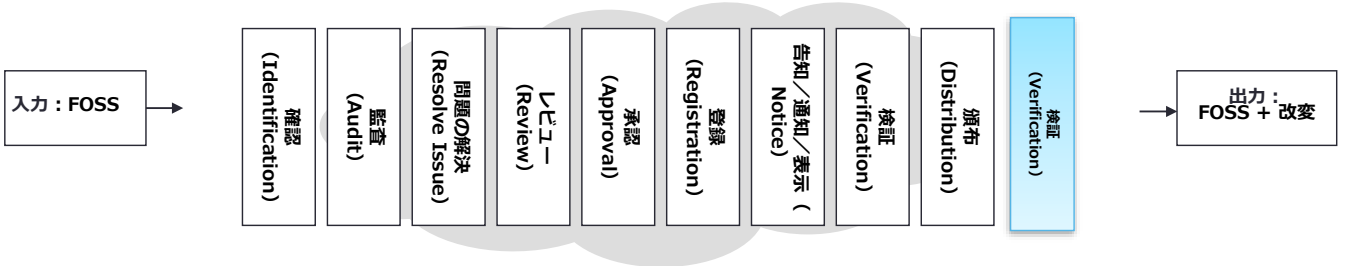
- ステップ：
  - 製品に対応したソースコードを関連ビルドツールや文書類とともに提供する（例：頒布Webサイトへアップロードする、頒布パッケージに含める）
  - ソースコードが、製品とバージョンに対応したラベルで識別される
- 成果：
  - ソースコードを提供する義務が履行される

※製品に対応したソースコードのこと

ソースコードを入手可能にする際、企業は製品に対応したソースコードをFOSSライセンスが許可する仕組みで提供します。このことは、ソースコードをソフトウェアの頒布にともに提供、またはそれを書面による申し出を通じ入手可能とすること、もしくはWebサイトでソースコードのアーカイブを公開することを意味します。

In cases where source code must be made available, the company provides the accompanying source code through the mechanisms permitted under the FOSS license. This may mean providing the source code along with the software distribution, making it available through a written offer, or posting a source code archive on a website.

最終検証



ライセンス義務のコンプライアンスを検証する

- ステップ：
  - 添付ソースコードが（あるならば）適切にアップロードされたか、または頒布されたかを検証する
  - アップロードされた、または頒布されたソースコードが承認されたものと同じバージョンとなっていることを検証する
  - 告知/通知/表示が適切に公開され、入手可能となっているかを検証する
  - その他確認された義務が履行されているかを検証する
- 成果：
  - 検証済みの頒布コンプライアンス関連資料が適切に提供される

このステップでは、企業の頒布行為がFOSSライセンスの義務を履行していることを検証します。このステップは一組織体としてFOSSレビュープロセス全体を監督する機能になりえるものです。

In this step, the company verifies that its distribution complies with its FOSS license obligations. This step could be a function of an entity providing oversight for the overall FOSS review process.

理解度チェック

- コンプライアンスの適正努力（Compliance due diligence）としてどのようなものが関係しますか？（本カリキュラムのプロセス例に挙げた各ステップについて概要を述べてください）
  - 確認
  - ソースコードの監査
  - 問題の解決
  - レビューの実施
  - 承認
  - 登録／承認の追跡
  - 告知／通知／ 表示
  - 頒布前の検証
  - 添付ソースコードの頒布
  - 検証
- アーキテクチャ レビューではどういったことを期待しますか？

本カリキュラムのプロセス例では以下のステップがありました。

- 確認（Identification） ー FOSSの使用を確認し追跡します。この作業はエンジニアからの要求、サードパーティによる開示、もしくはコード スキャンを通じて発生します。
- ソースコードの監査ー確認されたFOSSコンポーネントをライセンスと起源についてレビューします。
- 問題を解決するー FOSSポリシーに反したFOSSの使用を除去します。
- レビューの実施ーFOSSの使用に対する義務を査定し決定します。
- 承認ー承認の条件とライセンスの義務を明らかにします。
- 登録／承認の追跡ーその後のステップのために承認の条件とライセンス義務を追跡します。
- 告知／通知／表示ーFOSSライセンスで求められる形で告知文を準備します。
- 頒布前の検証ー頒布物のリリース前にコンプライアンスをレビューします。
- 添付ソースコードの頒布ーソースコードを必要に応じて入手可能にします。
- 検証ーコンプライアンス プロセスの監督を実施します。

アーキテクチャ レビューではFOSSコンポーネントと企業のソフトウェア間の関係を検査します。たとえば、FOSSと企業のコンポーネントがどのように互いにリンクするかといったことを検査します。

For our example process, the steps include:

- Identification - Identify and track FOSS usage. This may take place through engineer requests, third party disclosures, or code scanning.
- Auditing source code - Review identified FOSS components for license and origin information.
- Resolving issues - Remove FOSS usage that is incompatible with FOSS policies.
- Performing reviews - Assess and determine obligations for FOSS usage.
- Approvals - Communicate approval conditions and license obligations.
- Registration/approval tracking – Track approval conditions and license obligations for later compliance steps.
- Notices - Prepare notices as required by FOSS licenses.
- Pre-distribution verifications – Review distributions for compliance before release.
- Accompanying Source Code Distribution – Make source code available as needed.
- Verification – Provide oversight for compliance process.

Architecture reviews examine the relationships between FOSS components and company software. For example, how are FOSS and company components linked together?

第7章

コンプライアンスでの落とし穴とその回避

コンプライアンスの落とし穴

本章では、コンプライアンス プロセスで回避すべき潜在的な落とし穴について説明する。

1. 知的財産（IP）に関する落とし穴
2. ライセンス コンプライアンスに関する落とし穴
3. コンプライアンス プロセスに関する落とし穴

本章ではFOSSコンプライアンスプロセスで避けるべき、共通的な落とし穴について説明します。

In this chapter, we will describe some common pitfalls to avoid in the FOSS compliance process.

知的財産に関する落とし穴

タイプと説明	発見のされ方	回避策
コピーレフト型のFOSSがプロプライエタリ コードやサード パーティのコードに意図せずに取り込まれてしまう： このタイプの失敗は、開発プロセスにおいてエンジニアが、FOSSポリシーに反し、プロプライエタリにすることを意図したソース コードにFOSSコードを追加する時に起こる。	このタイプの失敗は、ソースコードをスキャンや監査実施の結果として、以下と合致可能性があるものとして発見される： • FOSSのソースコード • 著作権表示 ソースコード自動スキャン ツールはこの目的のために使用することができる。	このタイプの失敗は以下の対策によって回避できる： • コンプライアンス上の問題、各種タイプ／カテゴリーのFOSSライセンス、およびプロプライエタリ ソースコードにFOSSソースコードを取り込むことの意味をエンジニアリング部門のスタッフにトレーニングとして提供する • ビルド環境においてすべてのソースコードに対し、定期的にソースコード スキャンや監査を実施する

このスライドで挙げている最初の落とし穴は、コピーレフト型のライセンスのFOSSが気づかれずにプロプライエタリのコードと混在してしまうところで生じます。

この状況はライセンスの告知／通知／表示に関してソースコードを監査することや、コード スキャン ツールの使用を通じて発見されることがあります。

予防策として、エンジニアリング スタッフへのトレーニング提供、および開発プロセスにおける監査やスキャンの定期的な実施などがあります。

The first pitfall described in this slide arises where copyleft-style licensed FOSS is inadvertently mixed with proprietary code.

This may be discovered through auditing source code for license notices or using code scanning tools.

Preventative measures include training of engineering staff, and building regular audits or scans into the development process.

知的財産に関する落とし穴

タイプと説明	発見のされ方	回避策
コピーレフト型のFOSSとプロプライエタリ ソフトウェアのソースコードが意図せずリンクされてしまう このタイプの失敗は、 ライセンスが相互に矛盾するか両立しないソフトウェアをリンクした結果起こる。リンクの法的効果についてはFOSSコミュニティで議論の対象となる。	このタイプの失敗は異なるソフトウェア コンポーネント間の リンクに対し依存性追跡ツールを使うことで発見できる。	このタイプの失敗は以下の対策によって回避できる： 1. エンジニアリング スタッフをトレーニングし、FOSSポリシーの法的見解に反したライセンスを持つソフトウェア コンポーネントへリンクすることを回避する 2. ビルド環境全体に対し、継続的に依存性追跡ツールを実行する
ソースコードの改変を通じて プロプライエタリのコードがコピーレフト型のFOSSに組み込まれてしまう	このタイプの失敗は、FOSSコンポーネントに組み入れたソースコードを確認・分析するための監査やスキャンによって発見されることがある。	このタイプの失敗は以下の対策によって回避できる： 1. エンジニアリング スタッフへのトレーニング 2. 定期的なコード監査の実施

このスライドで挙げている最初の落とし穴は、コピーレフト型のライセンスのFOSSが気づかれることなくプロプライエタリ ソフトウェアにリンクされてしまうところで生じます。

このタイプの失敗は、依存性追跡ツールの使用や、アーキテクチャのレビューによって検出できます。

予防策は、エンジニアリング スタッフのトレーニングや、開発プロセスへのアーキテクチャ レビューの組み込みなどです。

2つ目の落とし穴は、プロプライエタリ コードがコピーレフト型ライセンスのFOSSに組み込まれることで生じます。たとえば、エンジニアリング チームがFOSSコンポーネントに対して行った改変により、プロプライエタリ コードが含まれてしまうようなケースです。

このタイプの失敗は、FOSSコンポーネントに組み込まれたソースコードを監査することで発見できます。

予防策としては、エンジニアリングスタッフのトレーニングや、開発プロセスへの定期的な監査を組み込まれることなどがあります。

The first pitfall in this slide arises where copyleft-style licensed FOSS is inadvertently linked to proprietary code.

This type of failure may be detected using dependency tracking tools or reviews of architecture.

Preventative measures include training of engineering staff, and building architectural reviews into the development process.

The second pitfall arises where proprietary code is included in copyleft-style licensed FOSS. For example, an engineering team making modifications to a FOSS component may include proprietary code in the modifications.

This type of failure may be discovered through auditing source code introduced into the FOSS component.

Preventative measures include training of engineering staff and building regular audits into the development process.

ライセンス コンプライアンスに関する落とし穴

タイプと説明	回避策
添付ソースコード／適切なライセンス、帰属表示、告知／通知を提供しない	このタイプの失敗は、製品を市場に出す前の段階で、ソースコードの全体像を捕捉し、製品のリリース サイクルごとのチェックリスト項目を公開することで回避できる。
間違ったバージョンのソースコードを提供してしまう	このタイプの失敗は、バイナリのバージョンに対応したソースコードが確実に公開されるようコンプライアンス プロセスに検証ステップを加えることで回避できる。
FOSSコンポーネントの改変に対応したソースコードを提供しない	このタイプの失敗は、FOSSコンポーネントに対応した原作のソースコードに加え改変に対応したソースコードが確実に公開されるようコンプライアンス プロセスに検証ステップを加えることで回避できる。

このスライドで挙げている最初の落とし穴は、企業が製品のバイナリに対応したソースコードを提供する義務を負っている一方で、その履行ができていないところで生じます。

2つ目の落とし穴は、企業がソースコードを提供していても、頒布したバイナリと合致する正しい版名の提供ができていないところで生じます。

3つ目の落とし穴は、企業がFOSSコンポーネントを改変したにもかかわらず、改変した版のソースコードを公開できていないところで生じます。企業は、代わりに原作版のFOSSコンポーネントを公開してしまうことがあります。

いずれのケースにおいても、失敗はコンプライアンス プロセスに適切なステップを実行することで回避できます。たとえば、リリースされたバイナリに対応するソースコードは、バイナリ版と併せてソースコードの全体像を捕捉し、保存されることが必要です。バイナリのリリースに合ったソースコードが確実に提供されるように、リリースに先立った検証作業でもチェックするべきでしょう。

The first pitfall in this slide arises where a company has an obligation to provide accompanying source code, but fails to do so.

The second pitfall arises where a company provides accompanying source code, but fails to provide the correct version that matches the distributed binary version.

The third pitfall arises where a company modifies a FOSS component, but fails to publish the modified version of the source code. The company instead publishes the source code for the original version of the FOSS component.

In each case, the failures may be prevented by properly applying steps in the compliance process. For example, source code for released binaries should be captured and stored along with the binary version. Verifications prior to release should check to ensure the proper source code is provided with the binary release.

ライセンス コンプライアンスに関する落とし穴

タイプと説明	回避策
FOSSソースコードの改変に印付けがされていない： 変更したFOSSのソースコードに、FOSSライセンスが要求する印付けがされていない（または改変に関する情報がライセンスを満足する上で十分なレベルの詳細さ、明確さとなっていない）。	このタイプの失敗は、以下の対策によって回避できる： 1. ソースコード リリース前の検証ステップでソースコード改変の印付けを行う 2. エンジニアリング スタッフにトレーニングを実施し、公開されるすべてのFOSSソフトウェアやプロプライエタリ ソフトウェアの著作権表示やライセンス情報をエンジニアリング スタッフが確実に更新できるようにする

このスライドで挙げている落とし穴は、企業がFOSSコンポーネントを改変する際、FOSSライセンスが求める改変への印付けをしていないところで生じます。この落とし穴は、コードに印付けするプロセスを実装したり、検証ステップの中で印付けしたりすることで回避できます。

The pitfall in this slide arises where a company modifies a FOSS component, then fails to mark its modifications when required by the FOSS license. This pitfall may be prevented through implementing processes for marking code or within verification steps.

コンプライアンス プロセスにおける失敗

説明	回避策	予防策
開発者がFOSSの使用について承認を求めない	このタイプの失敗はその企業のFOSSポリシーやプロセスに従事するエンジニアリング スタッフへの トレーニングの提供によって 回避できる。	このタイプの失敗は、以下の対策によって予防できる： 1. ソフトウェア プラットフォーム全体に対する定期的なスキャンを実施し、「宣言されていない」FOSSの使用を検出する 2. 企業のFOSSポリシーやプロセスに従事するエンジニアリング スタッフにトレーニングを提供する 3. 従業員の人事考課にコンプライアンスを含める
FOSSトレーニングが 受講されない	このタイプの失敗はFOSSトレーニングの修了を 従業員の専門性開発計画の一部とし、人事考課の管理対象にすることで回避できる。	このタイプの失敗は、指定期日までのFOSSトレーニング受講をエンジニアリング スタッフに義務付けることで予防できる 。

このスライドの落とし穴は、FOSSコンプライアンス プロセスがエンジニアリング チームに融合できないところから生じます。ここでは、エンジニアリング チームがFOSSの使用をレビュープロセスに上げない、もしくはFOSSの使用に取り組む方法についてトレーニングを受けないケースを挙げています。

予防策として、エンジニアリング トレーニングをモニタリングする、コンプライアンス プロセスをエンジニアリング チームに利用しやすいものにするといったことが挙げられます。

The pitfalls in this slide arise from a failure to integrate the FOSS compliance process with the engineering team. In these cases, the engineering team does not raise FOSS usage to the review process, or does not receive the training on how to handle FOSS usage.

Preventative measures include monitoring of engineering training, and also making the compliance process easily accessible to the engineering team.

コンプライアンス プロセスにおける失敗

説明	回避策	予防策
ソースコードの 監査が実施されない	このタイプの失敗は、以下の対策によって回避できる： 1. 周期的なソースコード スキャン／監査の実施 2. 定常的に監査を反復的開発プロセスにおけるマイルストーンと位置付ける	このタイプの失敗は、以下の対策によって予防できる： 1. スケジュール遅延とならないよう適切なスタッフを配置する 2. 定期的な監査を確実に実行する
監査で発見された 問題（スキャン ツールや監査レポートで「ヒット」したもの）を解決できない	このタイプの失敗は、監査レポートが未完了の場合にコンプライアンス チケットの解決（つまりクローズ）を許可しないことで 回避できる。	このタイプの失敗は FOSSコンプライアンス プロセスの承認ステップにブロック機能を実装することで予防できる。
FOSSレビューがタイムリーに求められない	このタイプの失敗は、エンジニアリング チームがFOSSソースコードの採用を決定していない場合でも、それより早期にFOSSレビュー リクエストを開始することで回避できる。	このタイプの失敗は教育を通じて予防できる。

このスライドでは、コンプライアンス プロセスの失敗によって発生する結果について述べています。最初は、FOSSコード ベースが開発の中で使用され、適切なレビューなしでリリースされるケースです。2つ目は、FOSSの使用は周知されていても、ライセンスの義務がレビュー・決定されていないケースです。最後は、コンプライアンス プロセスがリリース期限のプレッシャーに直面し、タスクを実行する時間が限られているケースです。

This slide describes potential consequences of compliance process failures. In the first case, a code base may be used in development and releases without proper review. In the second case, FOSS usage may be known, but license obligations are not reviewed or determined. In the last case, the compliance process may face release deadline pressures and have limited time to perform its tasks.

製品出荷前にコンプライアンスを確認する

- 企業は製品が （どのような形態であれ） 出荷される前にコンプライアンスを優先して実行しなければならない
- コンプライアンスを優先することで以下が促進される：
  - 組織内でのFOSSの効果的な使用
  - FOSSコミュニティやFOSS関連組織とのより良い関係

本章で述べた落とし穴を避けるには、リソースと努力が必要になりますが、FOSSコンプライアンス プロセスを優先することは重要なことです。そうすることで、開発プロセスにおけるFOSSの使用を効果的なものにし、またFOSSコミュニティにおける良好な協働関係の維持にも役立つことになります。

While avoiding the pitfalls described in this chapter may take resources and effort, prioritizing the FOSS compliance process is important. It can help you more effectively use FOSS in your development process, and also help maintain good working relationships within the FOSS community.

コミュニティとの関係を確立する

FOSSを商用製品に使用する企業として、FOSSコミュニティと良好な関係を創出し、維持することは非常によいことです。自身が使用し、商用製品にデプロイしているFOSSプロジェクトに関連する特定のコミュニティについては特にそうでしょう。

さらに、FOSS関連組織や団体との良好な関係は、コンプライアンスを履行する最良の方法について助言を得る上で、大いに助けになるでしょう。また、コンプライアンス上の問題についても助けてくれるでしょう。

ソフトウェア コミュニティとの良好な関係もまた、双方向コミュニケーションに役立つことでしょう。 （たとえばソフトウェアの改良をアップストリームに提供し、コミュニティのソフトウェア開発者からサポートを受けるといったこと）

FOSSコンプライアンス プロセスは、FOSSコミュニティにおける良好な協働関係を確立するための重要な要素です。

Your FOSS compliance process is a building block to establishing good working relationships within the FOSS community.

理解度チェック

- FOSSコンプライアンスではどのようなタイプの落とし穴がありますか？
- 知的財産に関する失敗例を1つ挙げてください。
- ライセンス コンプライアンスでの失敗例を1つ挙げてください。
- コンプライアンス プロセスでの失敗例を1つ挙げてください。
- コンプライアンスを優先することのメリットにはどのようなものがありますか？
- コミュニティとの良好な関係を維持するメリットにはどのようなものがありますか？

落とし穴は、次に大別されます： 知的財産（IP）における失敗 、ライセンス コンプライアンスでの失敗、コンプライアンス プロセスでの失敗

IPでの失敗の例は、プロプライエタリ コードとオープンソース コードの混合です。これは企業が望まない形でプロプライエタリ ソフトウェアを一般公開させる結果になりかねません。

ライセンス コンプライアンスでの失敗の例としては、オープンソース ソフトウェアの改変部に印付けすることを怠る、そのソフトウェアに含まれるオープンソース ソフトウェア コンポーネントを適切に記載していない、もしくはそのソフトウェアに対応するすべてのソースコードを入手可能にしていない、などがあります。

コンプライアンス プロセスでの失敗例として、オープンソース ソフトウェアの監査、レビュー、承認に関わるプロセスの失敗があります。監査者が、レポート中の全警告アイテムを「放棄した（Waived）」、またはレビューや承認プロセスに時間がかかりすぎた、などです。

コンプライアンスを優先することのメリットには、FOSSの使用をより効果的なものにできることや、 オープンソース コミュニティと良好な関係を構築できるといったことがあります。

コミュニティとの良好な関係を維持するメリットには、FOSSライセンスの要求への対応をよりよく評価できるようになること、FOSSの使用とコントリビューションについてより良い双方向コミュニケーションが得られること、などがあります。

Pitfalls can occur under the following categories: IP failure, license compliance failure, and compliance process failure.

An example of IP failure would be commingling of proprietary code and open source code, which may result in making proprietary software available to general public despite company's preference.

An example of license compliance failure would be a failure to mark an open source software after modification or to properly list the open source software components in the software or to make the complete and corresponding source code available.

An example of compliance process failure would be a failure in the process related to audit, review, or approving the open source software. Auditors "waived" all the red-flagged items in a report, or that the review and approval process takes too long.

The benefits of prioritizing compliance are that you become more efficient in your use of FOSS, and that you build a better relationship with the open source community.

The benefits of maintaining a good community relationship are that you can better assess how you can comply with the FOSS license requirements, and you have a better two-way communication with regard to contribution and use of the FOSS.

第8章

開発者向けガイドライン

開発者向けガイドライン

- **質が高く、十分にサポートされるFOSSコミュニティからコードを選ぶ**
- **指導を求める**
  - 使用する各FOSSコンポーネントに対し正式な社内承認を求める
  - レビューされていないコードを内部のソースツリーにチェックインしない
  - 外部のFOSSプロジェクトへのコントリビューションに正式な社内承認を求める
- **既存のライセンス情報を維持する**
  - 使用するFOSSコンポーネントに対し、これに備わっているFOSSの著作権情報やその他ライセンス情報を削除するなど、どのような形であっても損なうようなことをしない。すべての著作権と許諾情報は、すべてのコンポーネントで手を加えられないままにする
  - FOSSコンポーネントの名前を変更しない。ただしFOSSのライセンスで求められる場合は除く（たとえば、改変されたバージョンに名前の変更を求める場合がある）
- **FOSSのレビュープロセスのためにFOSSプロジェクトの情報を集め、保持する**

本スライドは、質の高いコンプライアンスへのアプローチのための鍵となる開発者向けガイドラインを概説しています。

--
This slide outlines the key developer guidelines necessary for a high quality compliance approach.

コンプライアンス プロセスとしての要求事項を見込む

OPENCHAIN

- **制定されたFOSSポリシーを遵守するために求められる時間を作業計画に織り込む**
  - FOSSソフトウェア使用のための開発者ガイドラインに従う。特に、プロプライエタリもしくはサードパーティのソースコードに（もしくはその逆）取り込んだり（Incorporating）、リンクしたり（Linking）する場合は特に意識する
  - アーキテクチャ計画をレビューし、両立しないFOSSライセンスのコンポーネントが混在することを回避する
- **コンプライアンスの検証を常に更新する – すべての製品に対して –**
  - 製品一つ一つごとにコンプライアンスの検証を行う：1つの製品に対してFOSSパッケージの使用が承認されたことが、他の製品について承認されることを必ずしも意味しないため。
- **そして、FOSSの新バージョンへのアップグレードごとにそれを行う**
  - 同じFOSSコンポーネントの各バージョンがレビューされ承認されることを確かなものとする
  - あるFOSSパッケージのバージョンをアップグレードする時に、新しいバージョンのライセンスが以前に使われていたバージョンのライセンスと同じであることを確認する（バージョンのアップグレードでライセンスが変更される可能性がある）
  - FOSSプロジェクトのライセンスが変更された場合にコンプライアンスの記録がアップデートされ、新しいライセンスが矛盾を起こさないことを確かなものとする

本スライドはコンプライアンス プロセスの要求事項をどうやって先に見込んでおくか、という点について説明しています。

--
This slides explains how to anticipate compliance process requirements.

コンプライアンス プロセスを
すべてのFOSSコンポーネントに適用する

OPENCHAIN

- **外製（In-bound：外部から入ってくる）ソフトウェア**
  - サプライヤから納入されるソフトウェアにどういったFOSSが含まれるかを理解するためのステップをとる
  - 自製品に含まれるであろうすべてのソフトウェアに対し負うべき義務を評価する
  - ソフトウェア提供者から受領したソースコードは必ず監査する、もしくはその代わりに、ソフトウェア提供者に対し、あらゆるソースコードについてソースコード監査レポートを義務付ける企業ポリシーを策定する

本スライドは、コンプライアンス プロセスが自社に入ってくるFOSSコンポーネントに対しどのように適用でき、適用する必要があるかについて強調しています。

--
This slide emphasizes how a compliance process can and should apply to all FOSS components entering your company.

理解度チェック

- 開発者がFOSSを取り扱う際に実践すべき一般的なガイドラインをいくつか挙げてください
- FOSSのラインセンス ヘッダー情報を削除したり変更したりするべきでしょうか？
- コンプライアンス プロセスにおける重要なステップをいくつか挙げてください
- 以前にレビューしたFOSSコンポーネントの新しいバージョンが発生させうる新たな問題とは、どういったものでしょうか？
- 外製のソフトウェアについて考慮すべきリスクとは何ですか？

無償で自由に利用できる開発者向けコンプライアンス基礎教育はThe Linux Foundationから提供されております。詳しくはこちらから：
https://training.linuxfoundation.org/linux-courses/open-source-compliance-courses/ compliance-basics-for-developers

FOSSを取り扱う際に開発者が実践すべき一般的なガイドライン：
・質の高いFOSSコミュニティからコードを選定する
・指導を求める
・既存のライセンス情報を維持する
・FOSSのレビュープロセスのためにFOSSプロジェクト情報を集め、保持する

FOSSのラインセンス ヘッダー情報を削除したり変更したりするべきでしょうか？
いいえ、既存のライセンス情報は保持されるべきです。ソースコードの改変や追加に対応して、追加のヘッダ情報を記述することは可能です。（注：ライセンスによって、ドキュメント上の変更を要求するものもあります）

コンプライアンスプロセスで重要なステップ：
・開発者向けガイドラインに従う。特に、FOSSコードがプロプライエタリのコードにリンクされたり組み込まれたりするような場合は特に意識する
・（製品）サイクルの早期からすべてのFOSSについてレビューし、承認する
・アーキテクチャをレビューし、両立しないライセンスのコンポーネントの混在を回避する
・リリース前に、すべての製品、すべてのバージョンについてFOSSコンプライアンスを検証する
・FOSSの新しいバージョンに対してコンプライアンスをレビューする

以前レビューされたFOSSコンポーネントの新バージョンが新たなコンプライアンス上の問題を起こす可能性：
・FOSSコンポーネントの新しいバージョンに対してFOSSライセンスが変更される（ghostscriptの例 https://en.wikipedia.org/wiki/Ghostscript）
・新バージョンの依存関係に伴って導入される新しいソフトウェアモジュールが、追加の義務を作り出すことがある。これらのモジュールは、FOSS頒布物に組み入れられる場合もあれば、ビルド時にリンクされる場合もある

外製のソフトウェアについて考慮すべきリスクとは何か？
・外製ソフトウェアに明に組み込まれたFOSSについてのライセンス コンプライアンス
・外製ソフトウェアを他のFOSSないしはプロプラエタリソフトウェアと統合することで生じる潜在的なライセンス上の矛盾
・外製ソフトウェアに含まれている、明らかになっていない、もしくは知られていないFOSSの存在

--
General guidelines developers can practices when working with FOSS:
- Select code from high quality FOSS communities
- Seek guidance
- Preserve existing licensing information
- Gather and retain FOSS project information for your review process

Should you remove or alter FOSS license header information? No – existing license information should be preserved, additional header information can be added for modifications or additions to source code (note, some licenses require documenting changes) .

Important steps in a compliance process:
- Follow developer guidelines, especially for any FOSS code included in or linked to proprietary code
- Review and approve all FOSS early in the cycle
- Review architecture and avoid mixing components governed by incompatible licenses
- Verify FOSS compliance for every product and every version prior to release
- Review FOSS compliance for new versions of FOSS

A new version of a previously reviewed FOSS component can create new compliance issues by:
- A change in the FOSS license for the new version of the FOSS component(e.g. ghostscript https://en.wikipedia.org/wiki/Ghostscript)
- New dependencies introduced with new versions which create additional FOSS obligations. These dependencies may be embedded in the FOSS distribution or they may be dependencies resolved at build time.

What risks should you address with in-bound software?
- License compliance for any disclosed FOSS embedded in the in-bound software
- The potential for creating license conflicts by integrating inbound software with other FOSS or proprietary software
- Undisclosed or unknown FOSS included in the in-bound software