



OPENCHAIN

# 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 [Creative Commons CC0 1.0 Universal](https://creativecommons.org/licenses/by/4.0/) ライセンスの下でリリースされています。

本スライドの使用、改変および共有にあたっての制限はありません。

また、これらは無保証となります。

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

# 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>

# コンテンツ

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

# 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>

## 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

# 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



# ソフトウェアに最も関係する 著作権における「権利」

- 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

# 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、その所有者は、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

# 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ることができる
  - 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

# 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？
- 特許はその所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？

## 第2章

---

# FOSSライセンス概論

# FOSSライセンス

- 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/>

# 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

# 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



# 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

- 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

# 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

# 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

# ライセンスの両立性（互換性） ※

- 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がある。

# 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ことを知らせる
- **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

# マルチライセンス

- 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

# 理解度チェック

- FOSSライセンスとはどのようなものでしょうか？
- パーミッシブなFOSSライセンスの典型的な義務としてどのようなものがありますか？
- 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う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のようなものでしょうか
- FOSSの告知／表示にはどのような情報がありますか？またそれらはどのように使われますか？



## 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

- **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

# 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？

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

# 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

# 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

# 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

# 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

# コンプライアンスのメリット

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



# 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？

## 第4章

---

# FOSSレビューにおける ソフトウェアの重要概念

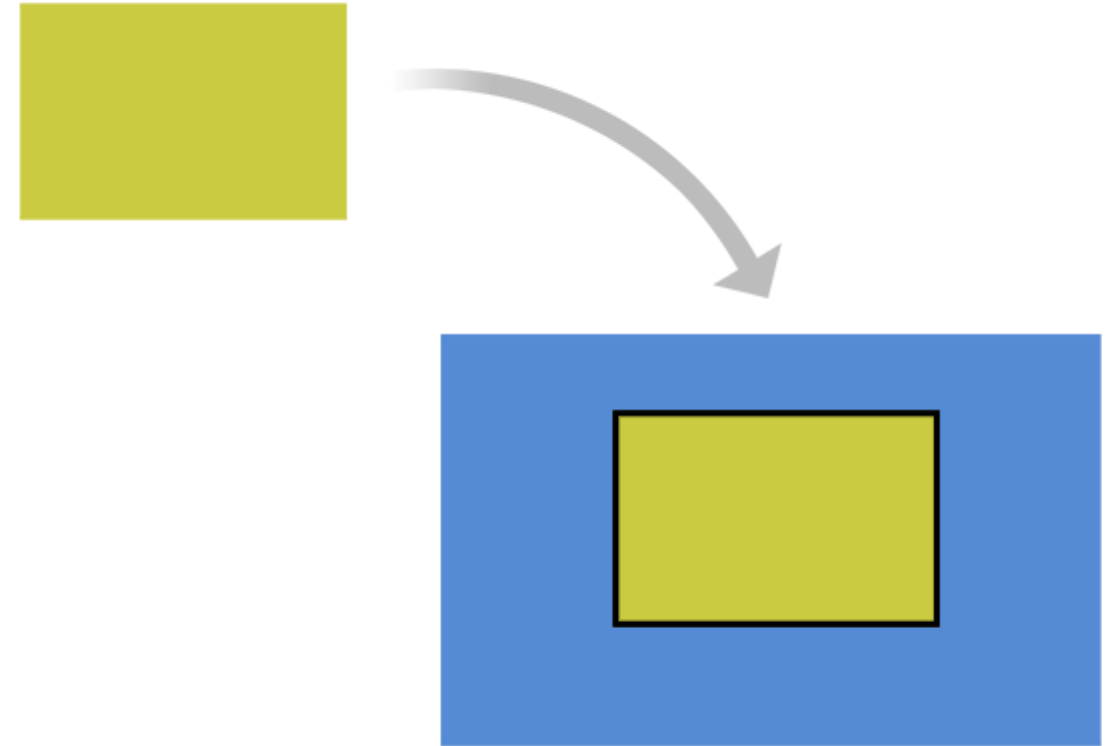
# そのコンポーネントをどう使うのか？

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)

# 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)

# 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)



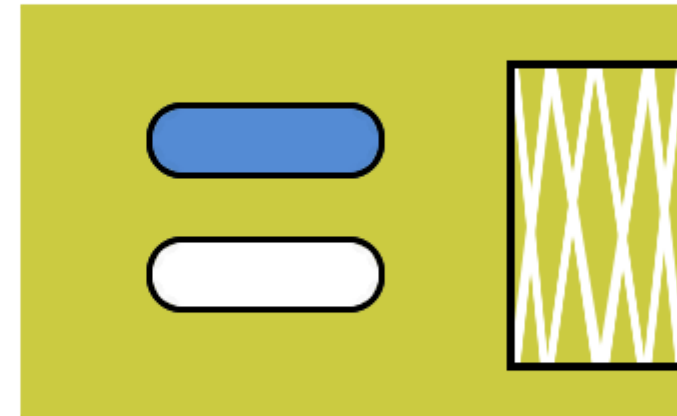
# 改変する (Modification)

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)

- 
- 追加
  - 注入

- 
- 削除



- 修正
- 最適化
- 変更

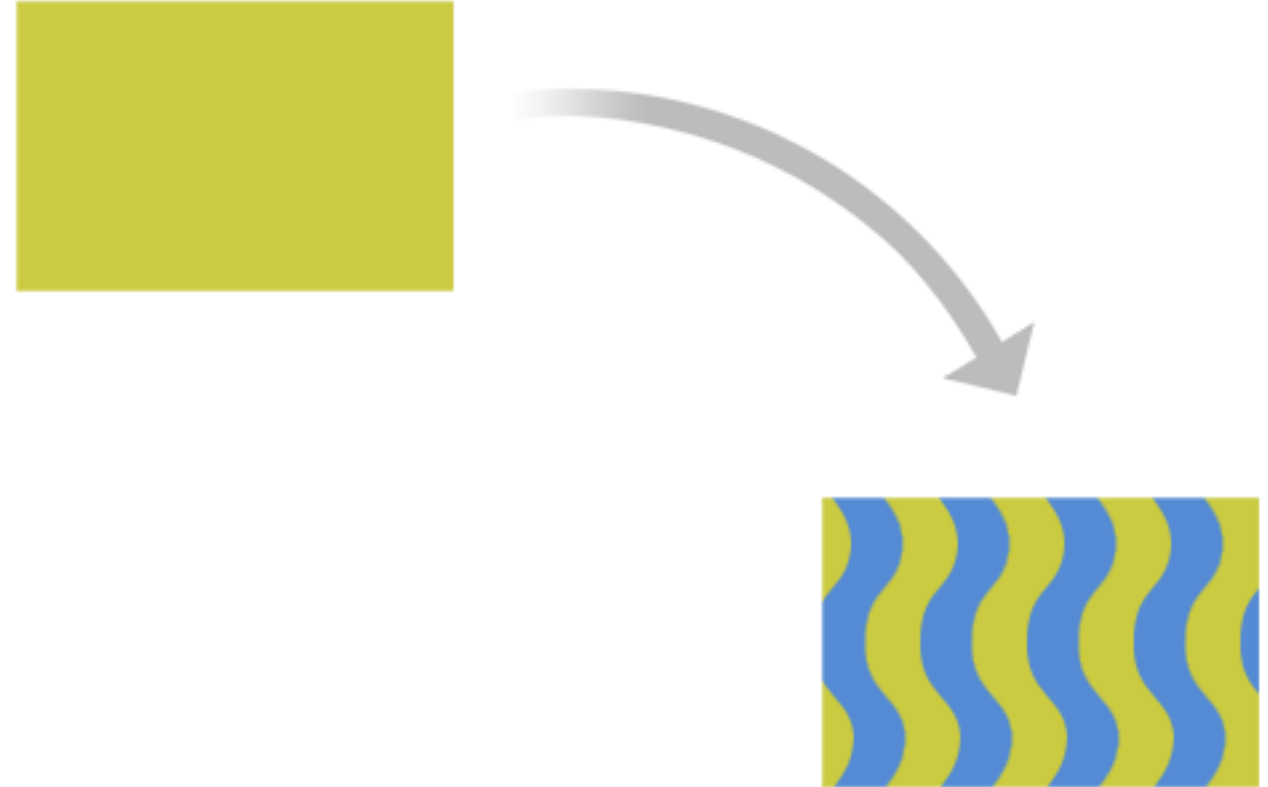


# 翻訳する (Translation)

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：

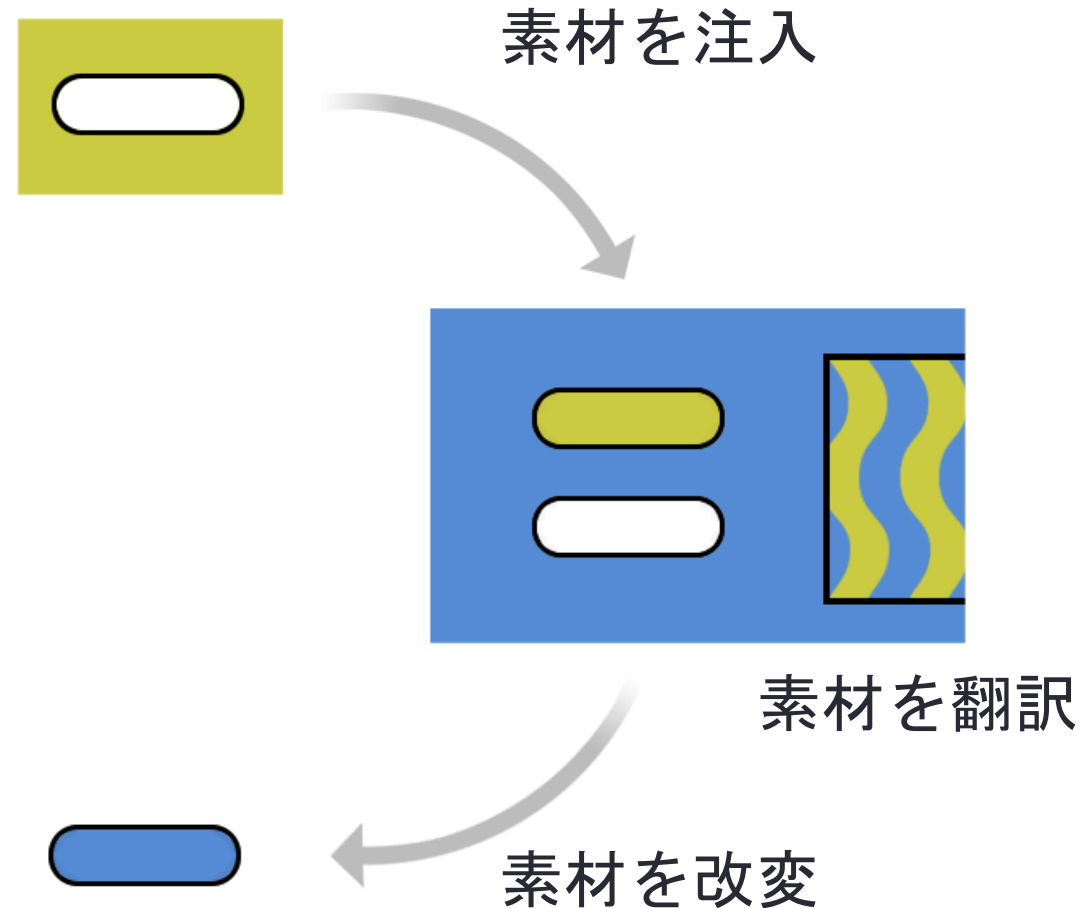
- 中国語から英語への翻訳
- C++ からJavaへの変換
- バイナリへのコンパイル



# 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。





# 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

# 理解度チェック

- 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？

## 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レビューの開始



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

# どのような情報を集める必要があるか？

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

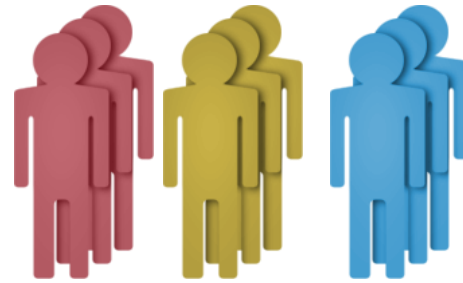
# FOSSレビューチーム



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）

# 提案されたFOSSの使用を分析する



法務      調査・分析      専門家

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：

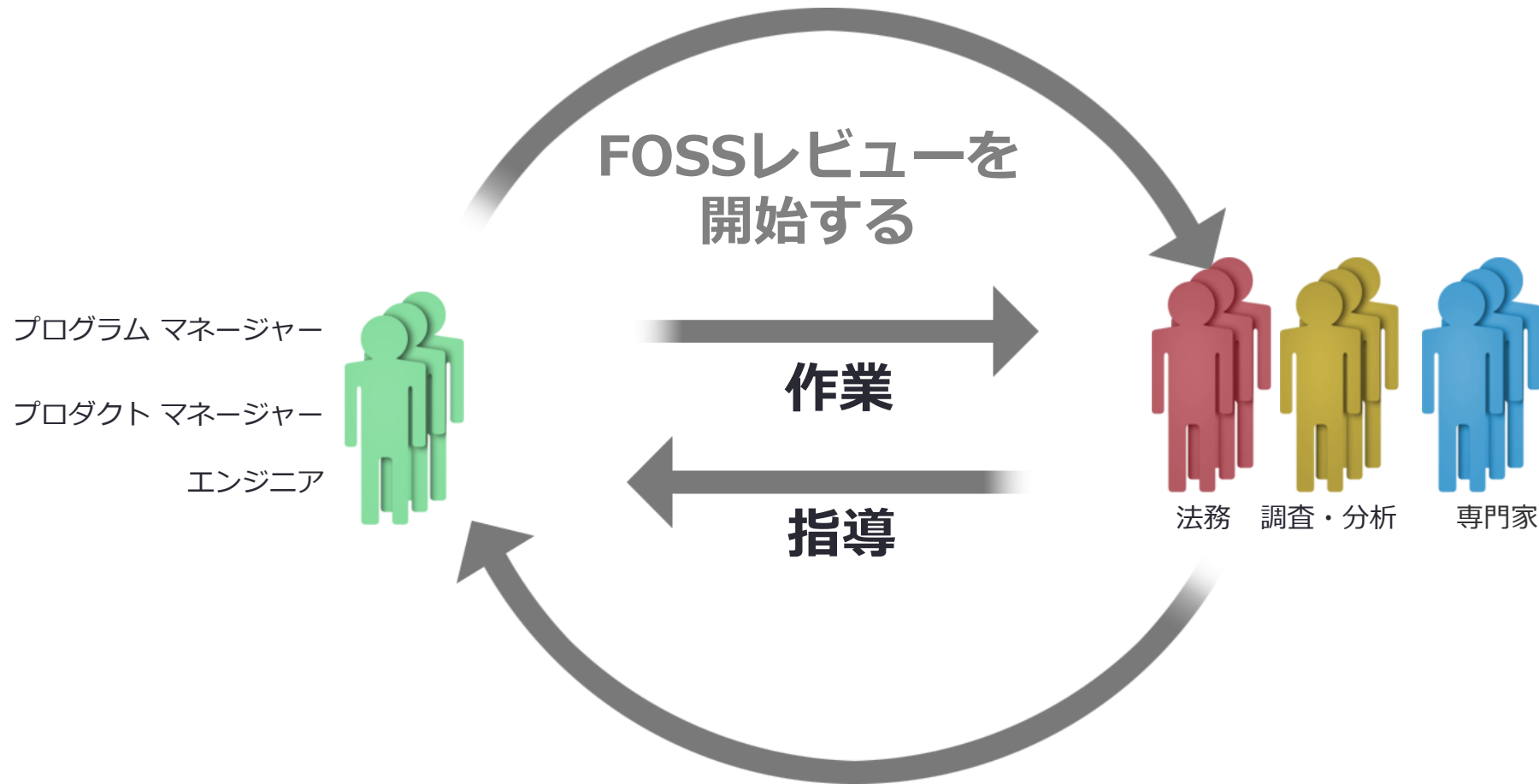
- コードと付随した情報が、完全で、一貫していて、正確か？
- 宣言されたライセンスがコードファイルにある内容と合致しているか？
- ソフトウェアを構成する他のコンポーネントとともに使用することをライセンスが許容しているか？



# ソースコード スキャン ツール

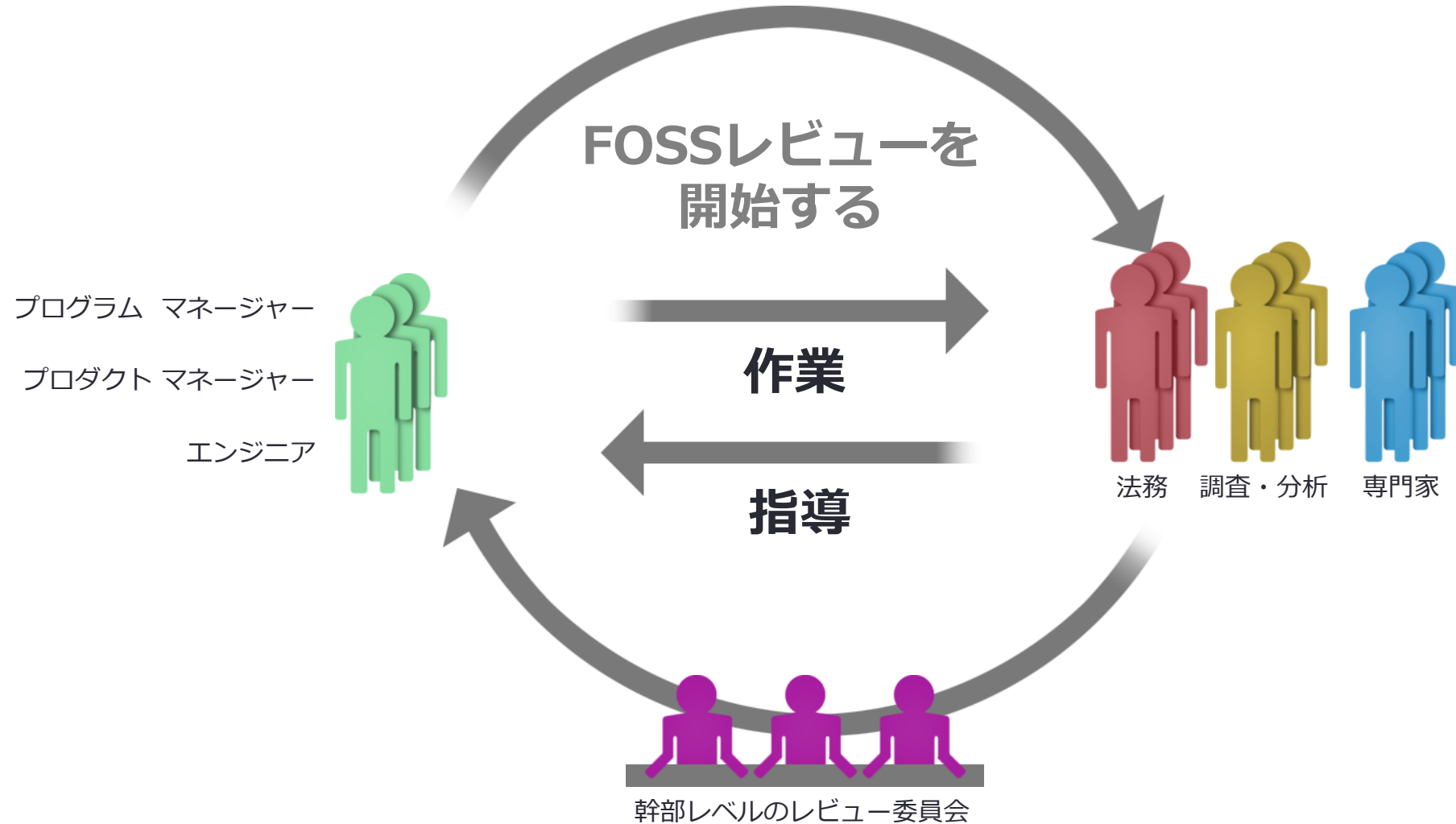
- ソースコードスキャンを自動化するツールは数多く、さまざまなものが存在
- それらのすべては特定のニーズに向けたソリューションであるため、あらゆる課題を解決するものではない
- 企業はそれらの中で自分たちの特定の市場領域や製品に合うものを選定する
- 多くの企業は自動化ツールと人手によるレビューを併用している
- 無償で、自由に利用できるソースコード スキャン ツールの1つのよい例としてThe Linux Foundation でホストしたプロジェクトである、FOSSologyがある：<https://www.fossology.org>

# 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認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

# 理解度チェック

- 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？

## 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



# 中・小規模企業のチェックリストの例

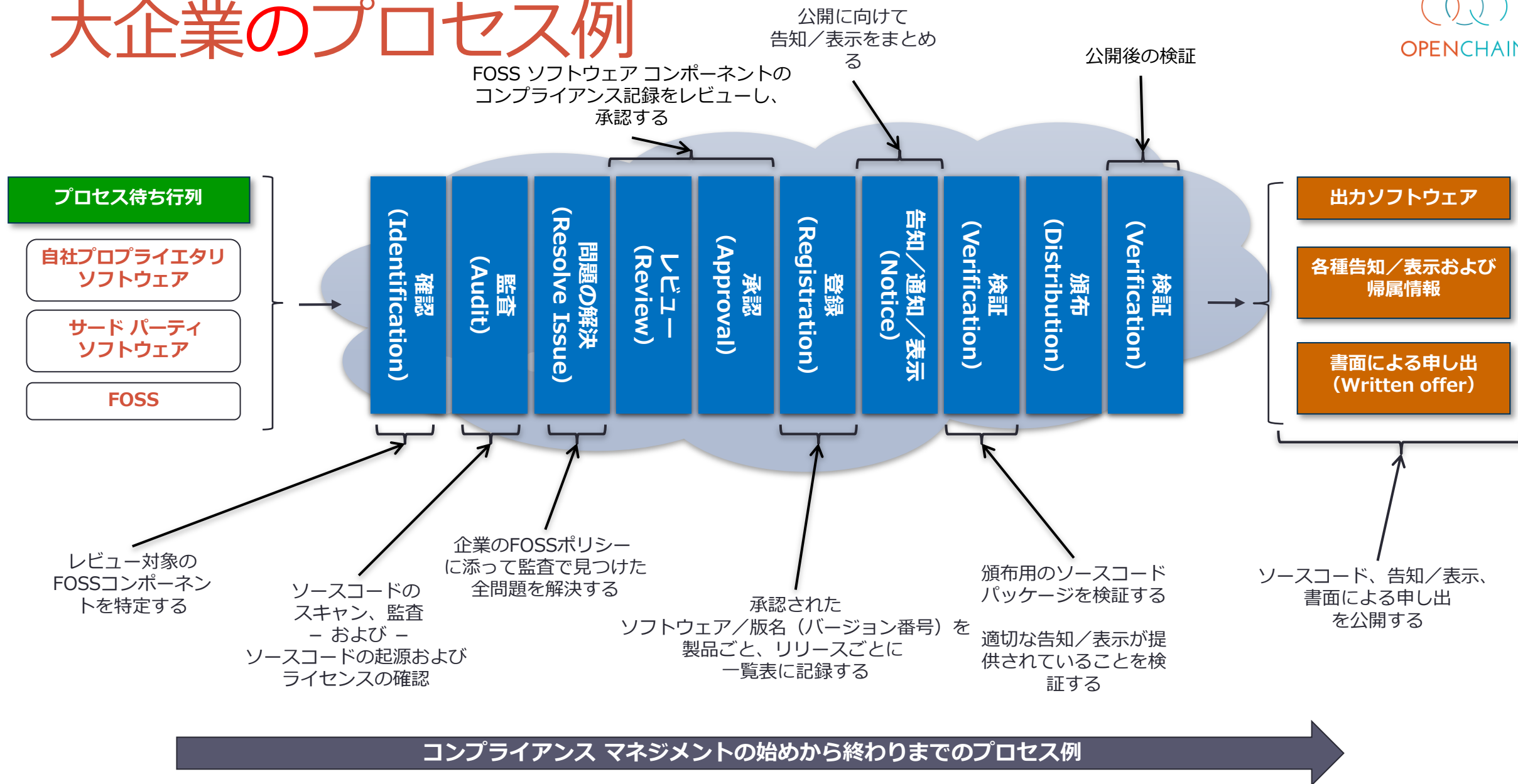
## 継続的に行うコンプライアンス関連タスク：

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：

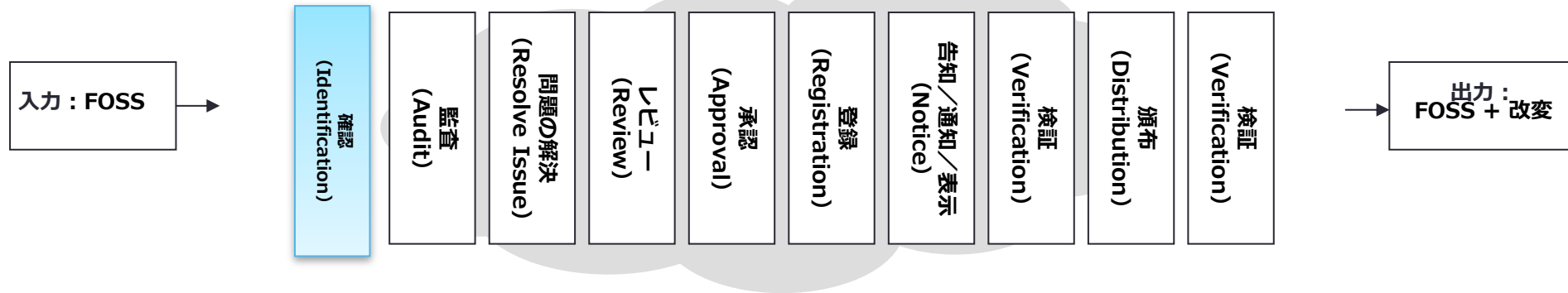
1. 適切なコンプライアンス対応要員を配置し、明確な責任ラインを指定する
2. 既存のビジネスプロセスをFOSSコンプライアンスプログラム支援のために適合させる
3. 組織のFOSSポリシーについて全社員が学習できるトレーニングを持つ
4. FOSSコンプライアンスに関係するすべての活動の進捗を追跡する

# 大企業のプロセス例





# 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

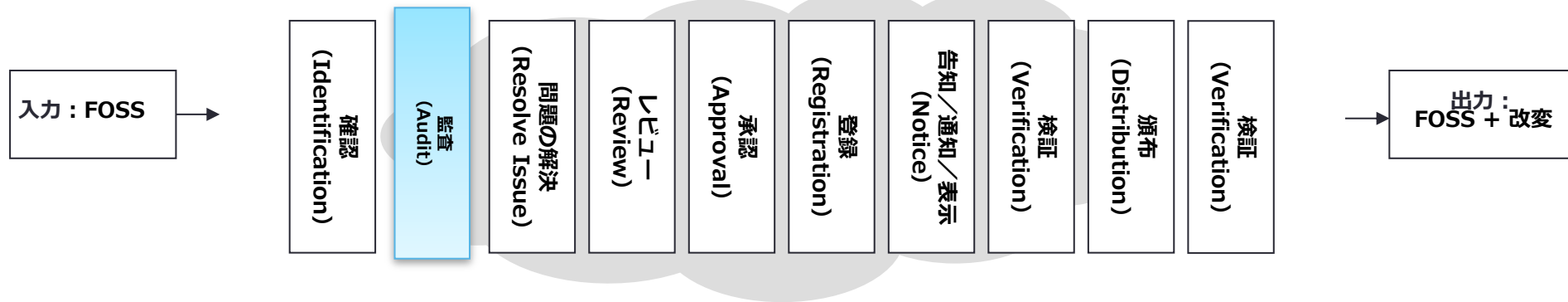
### • ステップ :

- 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

# 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

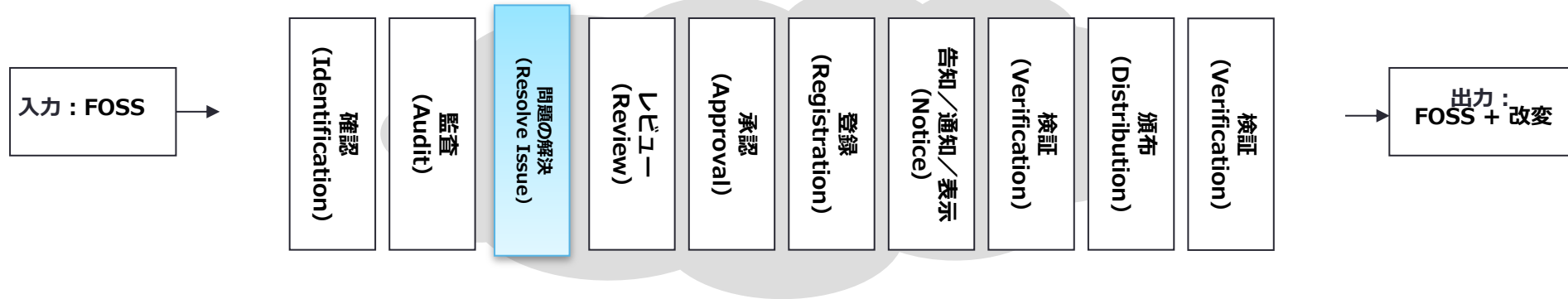
### • ステップ :

- 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を検証される
- 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

# 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）


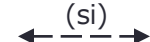
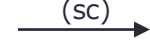
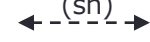
### • 成果 :

- 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

# アーキテクチャ レビュー (テンプレートの例)

## 凡例

-  プロプライエタリ
-  サードパーティの商用
-  GPL
-  LGPL
-  FOSS パーミッシブ

-  (fc) 関数呼び出し
-  (si) ソケット インターフェース
-  (sc) システム コール
-  (sh) 共通ヘッダー

ユーザー空間

[ コンポーネント名を記入してください ]

[ 相互作用の仕方を記入してください ]

カーネル空間

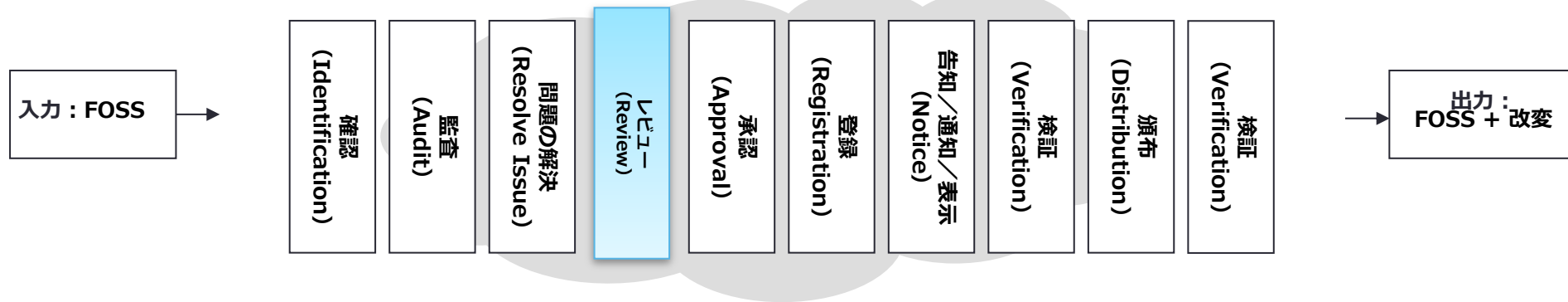
[ コンポーネント名を記入してください ]

[ 相互作用の仕方を記入してください ]

ハードウェア

[ コンポーネント名を記入してください ]

# 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 :

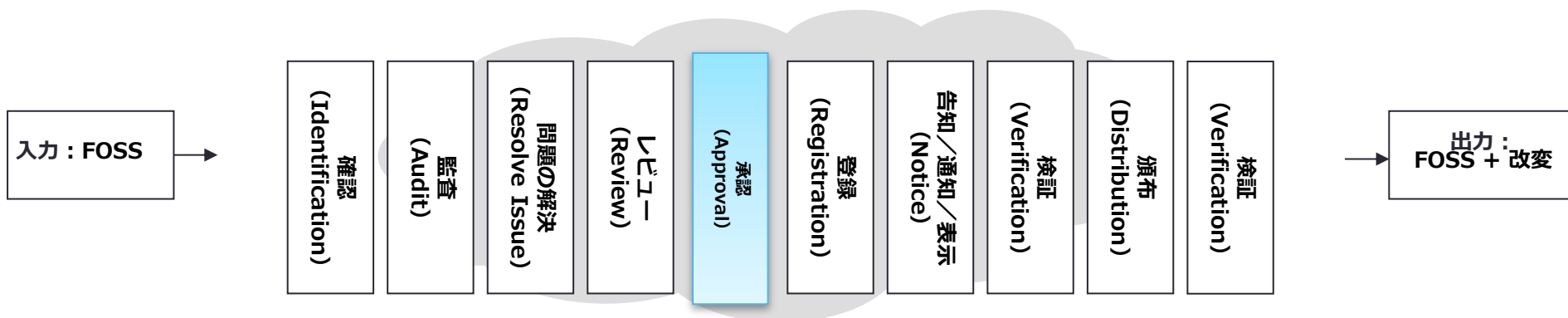
- レビュー スタッフに適切な職権レベルを含める
- FOSSポリシーに則ったレビューを実施する

- 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認なものとする
- 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

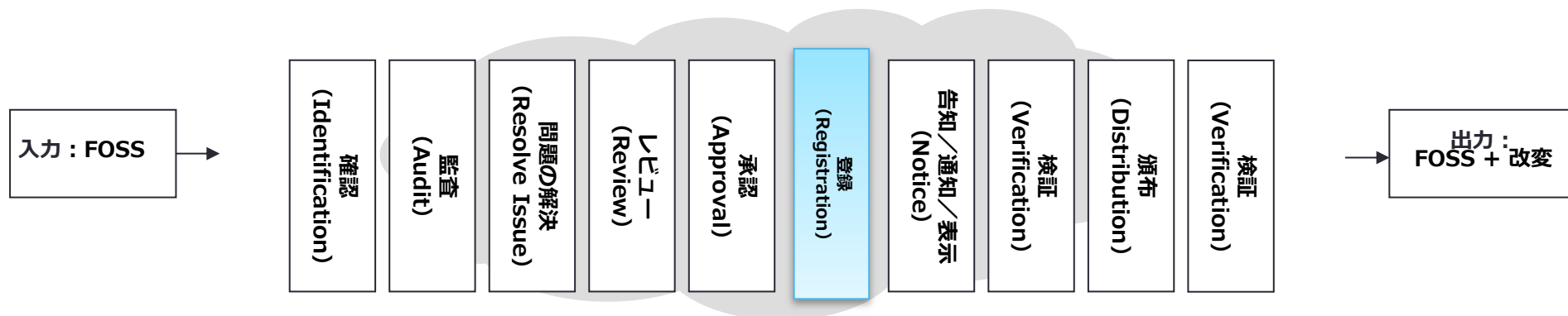
# 承認

- 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

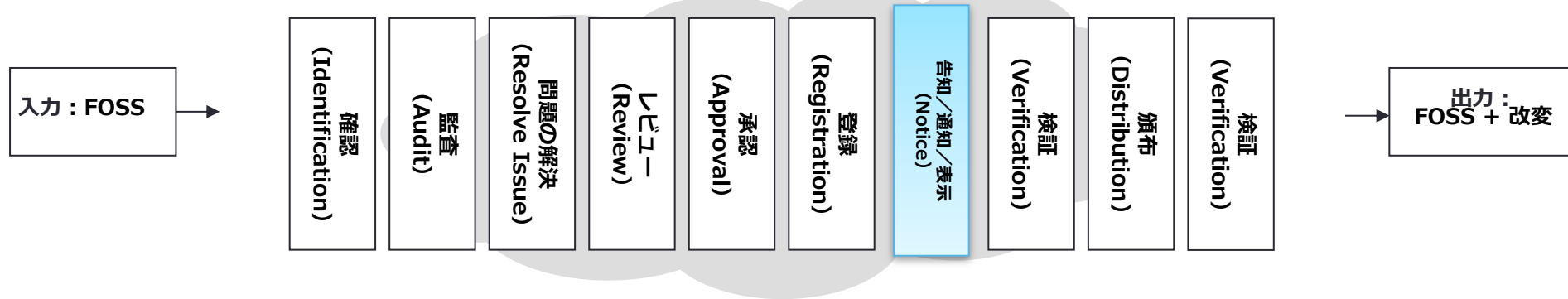


# 登録／承認の追跡

- 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



# 告知／通知／表示

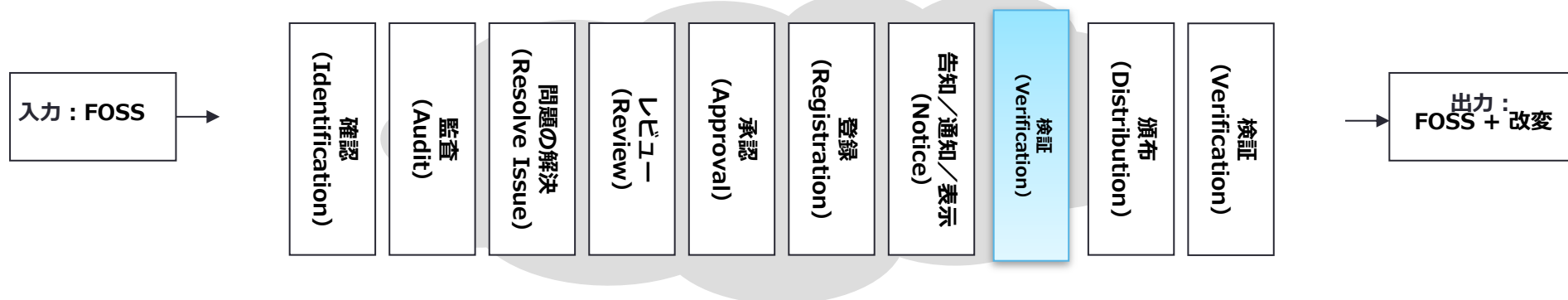


## 製品リリース時に用いる適切な告知／表示を準備する

- 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



# 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

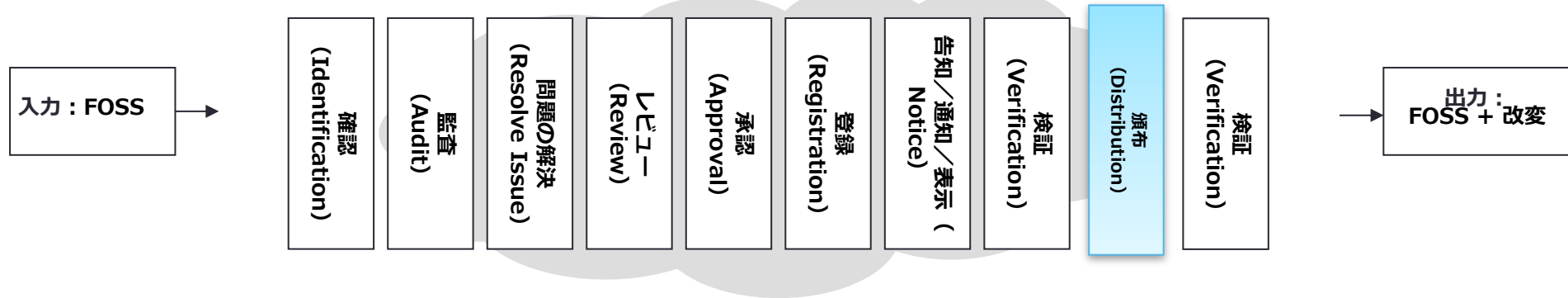
### • ステップ:

- 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

# 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:

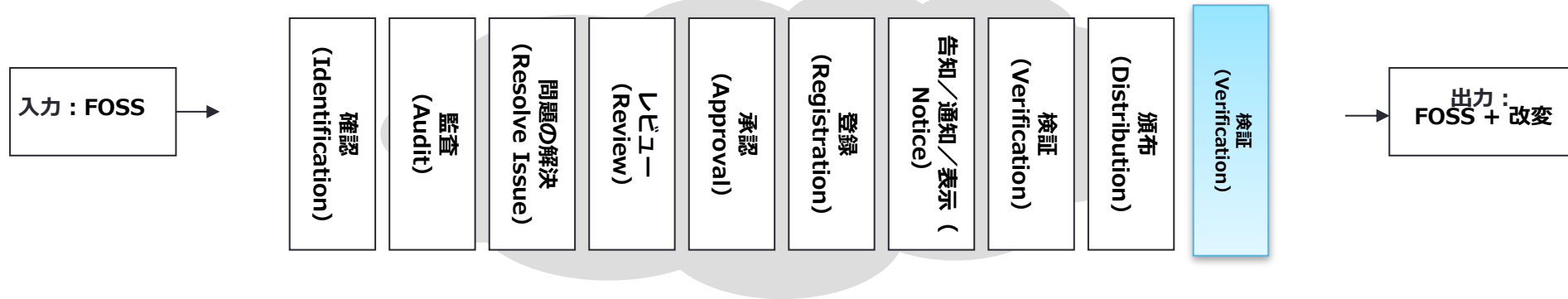
- 製品に対応したソースコードを関連ビルドツールや文書類とともに提供する（例：頒布Webサイトへアップロードする、頒布パッケージに含める）
- ソースコードが、製品とバージョンに対応したラベルで識別される

### • 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

# 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

# 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 レビューではこういったことを期待しますか？

## 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

# 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
<p><b>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：</b></p> <p>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。</p>	<p>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：</p> <ul style="list-style-type: none"><li>• FOSSのソースコード</li><li>• 著作権表示</li></ul> <p>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。</p>	<p>このタイプの失敗は以下の対策によって回避できる：</p> <ul style="list-style-type: none"><li>• 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</li><li>• 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</li></ul>

# 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
<b>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</b>  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
<b>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</b>	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



# 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。

# 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
<p><b>FOSSソースコードの改変に印付けがされていない：</b></p> <p>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）。</p>	<p>このタイプの失敗は、以下の対策によって回避できる：</p> <ol style="list-style-type: none"><li>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</li><li>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</li></ol>

# 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： <ol style="list-style-type: none"><li>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</li><li>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</li><li>3. 従業員の人事考課にコンプライアンスを含める</li></ol>
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。

# 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
<b>ソースコードの 監査が実施されない</b>	このタイプの失敗は、以下の対策によって回避できる： <ol style="list-style-type: none"><li>1. 周期的なソースコード スキャン／監査の実施</li><li>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</li></ol>	このタイプの失敗は、以下の対策によって予防できる： <ol style="list-style-type: none"><li>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</li><li>2. 定期的な監査を確実に実行する</li></ol>
<b>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</b>	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。
<b>FOSSレビューがタイムリーに求められない</b>	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。

# 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

# 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）

# 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？

## 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**

# コンプライアンス プロセスとしての 要求事項を見込む

- **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

# 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

- **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こういった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

# 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る新たな問題とは、こ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>