OPENCHAIN

カリキュラム

FOSSトレーニング参照資料 第2版 最終ドラフト
OpenChain 仕様書　第1.0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。

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「 OpenChain Curriculum Release 2 」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.

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

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1.0の1.1.1項）>>

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）：作者が著した原作を保護する
  - （根底のアイデアではなく）表現を保護
  - ソフトウェア、書物、音響・映像作品、半導体マスク（回路パターンの原版）
- 特許権（パテント）：新規性、有用性、非自明性を持つ発明
  - 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
- 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

著作権の中でソフトウェアに最も関係する「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※」 を作る権利- 修正を加えることができる
  - 派生的著作物という用語は、原作に基づいてはいるものの、その新しい著作物が原作のコピーではなく、原作に対し独自に創造的な作業が加えられたと主張できるレベルの作品を言う（この用語は米国法令に基づいているの要留意）
- 「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コミュニティの中に　　おいても、関連した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

ソフトウェアにおける特許の概念

- 特許は、機能を保護するーこれには、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
- 特許保有者は、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ことができる
- 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

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

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（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？

第2章

FOSSライセンス概論

FOSS（フリー／オープンソース ソフトウェア）ライセンス

- FOSSソフトウェアのライセンスは、一般的に改変と再頒布を許容する条件の下で、ソースコードの入手が可能となっている
- 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)」と言われる

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）を原作と同一の条件で再頒布することを要求するものがある
- これは、「コピーレフト」、「互恵的」、あるいは「遺伝的」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
- コピーレフト ライセンスは、ソースコードが入手できる状態にあることを義務付ける場合がある

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、もしくは再頒布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、多くの場合、金銭の支払いやライセンス料を伴う
- 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

その他のライセンス

- フリーウェアープロプライエタリ 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
- シェアウェアー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
- フリーウェアとシェアウェアは、FOSSではない

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」に対し義務を拡張している
  - 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

告知／表示

告知／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の一定の場所に告知／表示を設定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。

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の下でソフトウェアを頒布する手法
  - 例：ソフトウェアが「デュアルライセンス」である場合、受領者はそのソフトウェアの使用や頒布に際し、2つのライセンスのどちらかを選択できる
- 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

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？

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（FOSSの使用を検出し、追跡する）を認識すること。**自身のソフトウェアを構成するすべてのFOSSコンポーネント（および、それぞれで確認されたライセンス）を特定、追跡し、そのリストを保管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。**組織のコンプライアンス プログラムは、業務遂行上生じる代表的なFOSSユースケースを認識し、これに対応する必要がある

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？

関与するライセンスにもよるが、義務としては以下のようなものがある。

- **帰属やその他告知。** 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ように、ソースコードや製品の関連文書、もしくはユーザ インターフェース上に著作権やライセンスに係る文言を含めること。
- **ソースコードの提供。**原作ソフトウェア、組み込んだソフトウェアや改変部分、およびビルド用のスクリプトも含んだソースコードを提供すること。

以下を契機としてこれらの義務が発動する場合がある：

- 外部への頒布
- 改変を加えたかどうか

FOSSにおける条件と制約

使用するFOSSによっては以下の条件や制約のうち1つもしくはそれ以上に従う必要がある。

- 著作権表示（および、その他の告知）を保持すること
- ライセンスの写しを提供すること
- 改変告知を提供すること
- 混乱を避けるために、改変版の名前を異なる名前とすること
- （改変の有無を問わず）ソースコードへのアクセス先を提供すること
- 改変版（派生的著作物）を同じライセンス下に置くこと
- 帰属告知を提供すること
- プロジェクト名、著作権保有者名、商標を使用しないこと
- 原作のライセンスの下で供与された権利を他者に制限すること
- 解除条項（違反すれば、ライセンスを失うこと）

FOSSコンプライアンスのトリガー：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。その際のトリガーとは「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と相互に作用すること」。
  - いくつかのライセンスが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を含めたトリガー イベントを定義している。（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）

FOSSコンプライスのトリガー：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）
- 改変が派生的著作物を生み出し、FOSS の著作者が改変に対し義務を課したり制限したりすることもある
- 改変をトリガーとして発動されるFOSSの義務の例：
  - 改変の告知
  - 製品のバイナリに対応したソースコードの提供

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. 商用製品におけるFOSSの効果的使用を促進する
2. FOSS開発者の権利を尊重し、ライセンス義務を果たす
3. オープン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：

- 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
- FOSSコミュニティやFOSS関連組織とより良い関係が生まれる
- 有効なFOSSソリューションについての知識が高まる

理解度チェック

OPENCHAIN

- 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

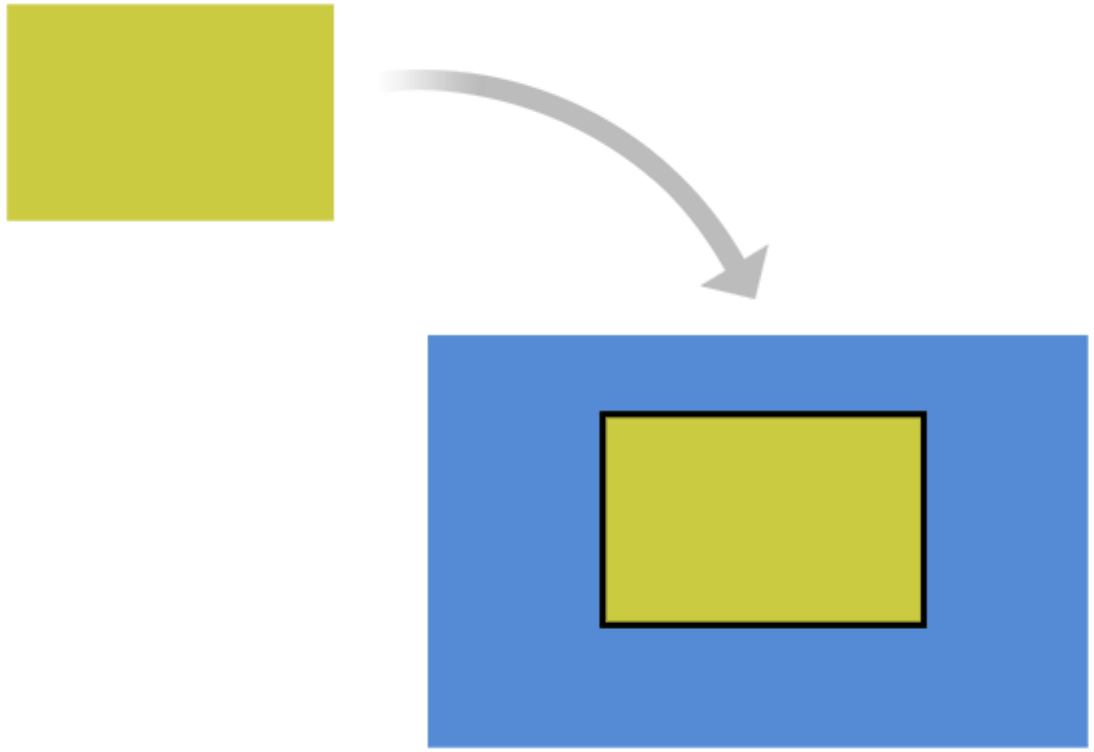
そのコンポーネントをどう使うのか？

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。

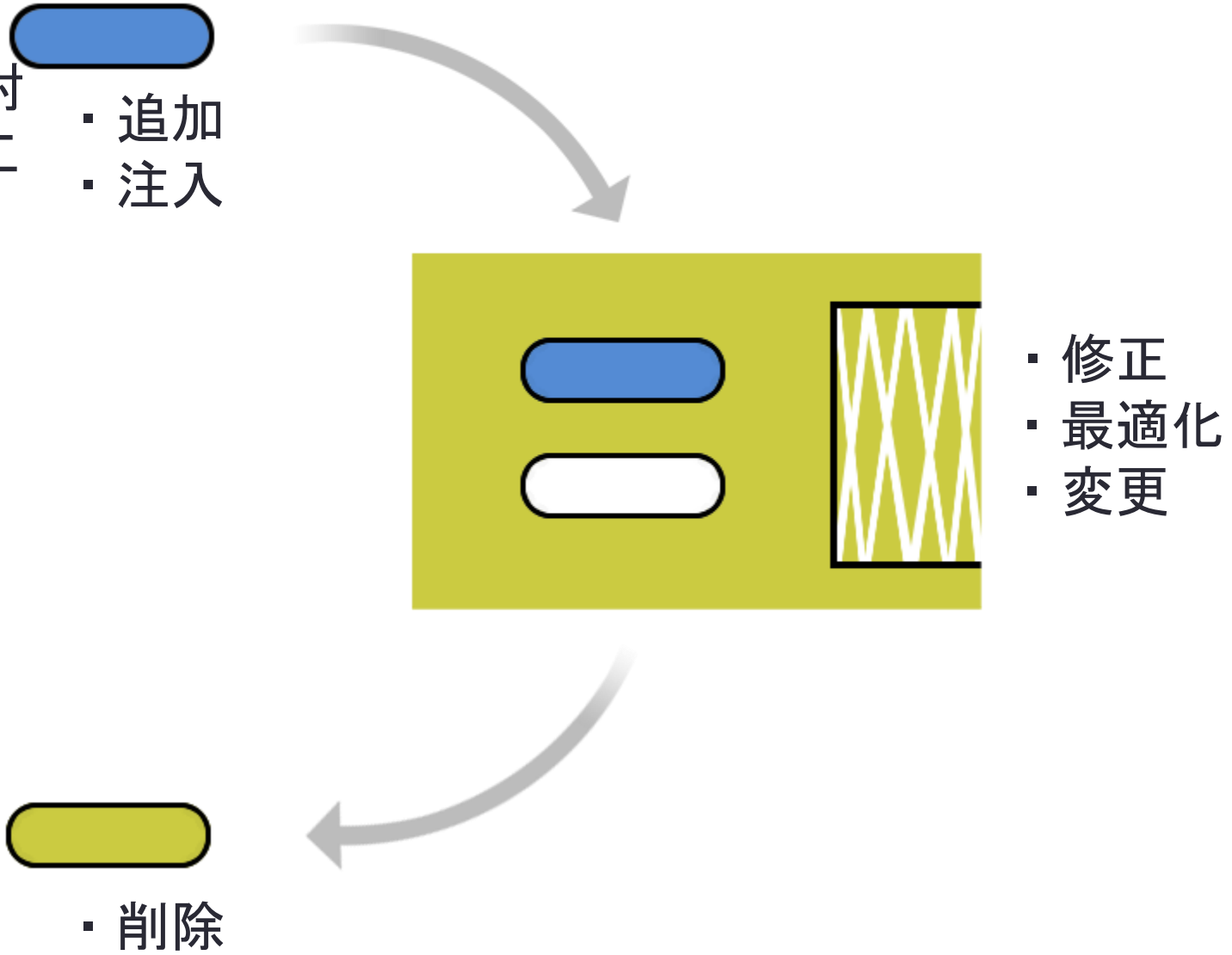
関連する用語：

- 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）

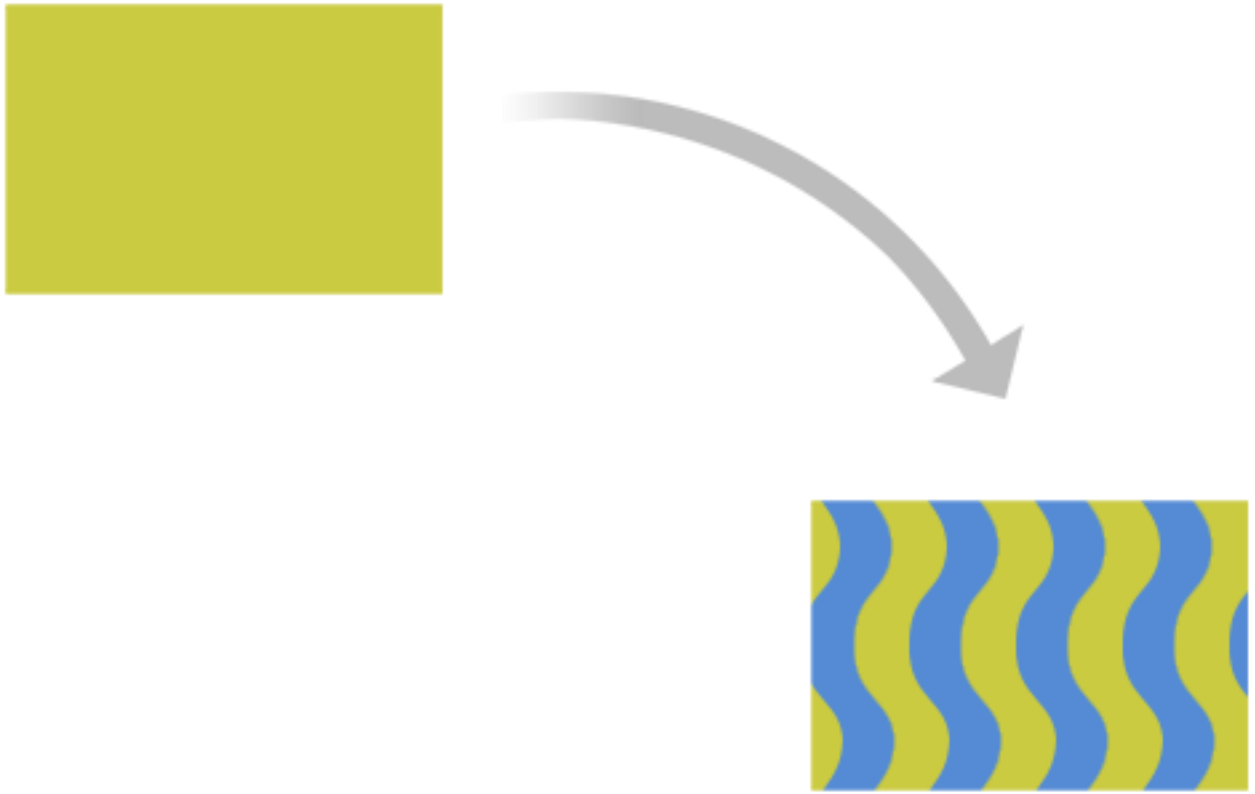
開発者はFOSSコンポーネントに対して、次のように変更を加えることができる：

- 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）



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。



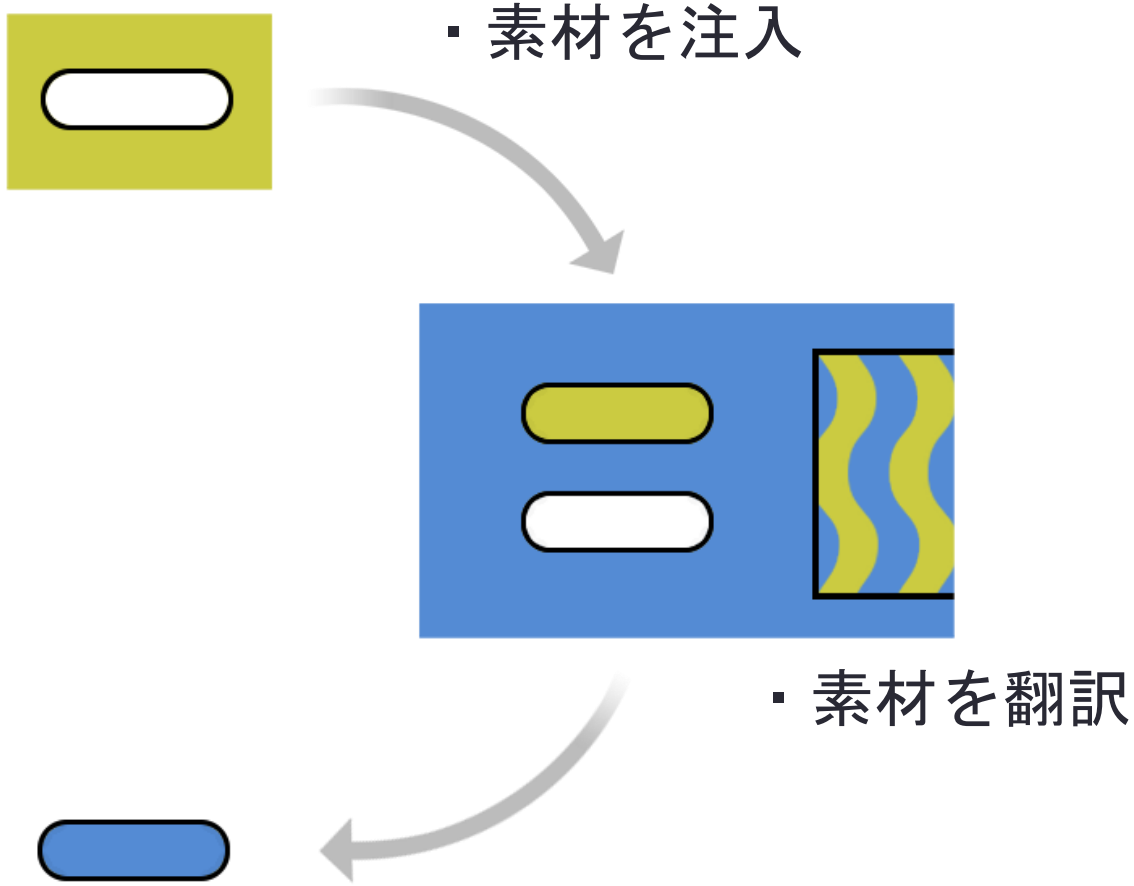
例として以下のようなものがある：

- 中国語から英語への翻訳
- C++ からJavaへの変換
- VHDLのマスクパターンやネットリストへのコンパイル
- 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

- 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

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

OPENCHAIN

- 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FOSSに関する義務を分析し決定することができる
- 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決定
  - 企業のポリシーや事業目標の観点からの指導

FOSSレビューの開始

FOSSレビューのプロセスは、FOSSを取り扱うプログラム マネージャー、プロダクト マネージャー、エンジニアなどの参加が必要。

注：このプロセスは外部ベンダーからFOSSベースのソフトウェアを受領した時に開始される場合もある。

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。

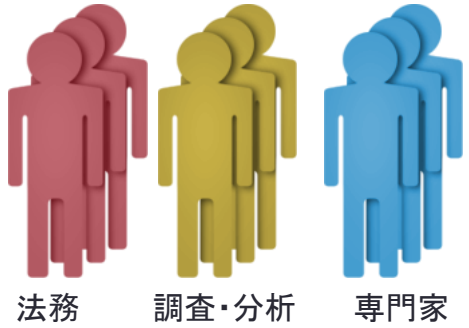
- パッケージ名
- 版名（バージョン番号）
- オリジナルのダウンロード元URL
- ライセンスおよびライセンスのURL
- 説明
- 改変に関する記述
- 依存関係のリスト
- 製品で意図している使用方法
- そのパッケージを内包する製品のファースト リリース（最初の公開・販売）
- ソースコードを入手できるか
- ソースコードがどこでメンテナンスされるか
- そのパッケージが他の経緯で以前に承認されたことがあるか？
- 輸出管理対象となる技術が含まれているか
- 外部ベンダーからの提供物の場合：
  - 開発チームのコンタクト ポイント
  - 著作権表示、帰属表示、およびライセンスの義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 レビューには複数の支援グループが参加し、FOSSの使用に関する支援、指導、とりまとめ、およびレビューを協力して行う。レビュー チームには、以下の複数のチームが含まれる。

- ライセンスの義務を特定し、評価する法務チーム
- FOSSの使用の確認と追跡を支援するスキャン・ツール サポート チーム
- 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専門家

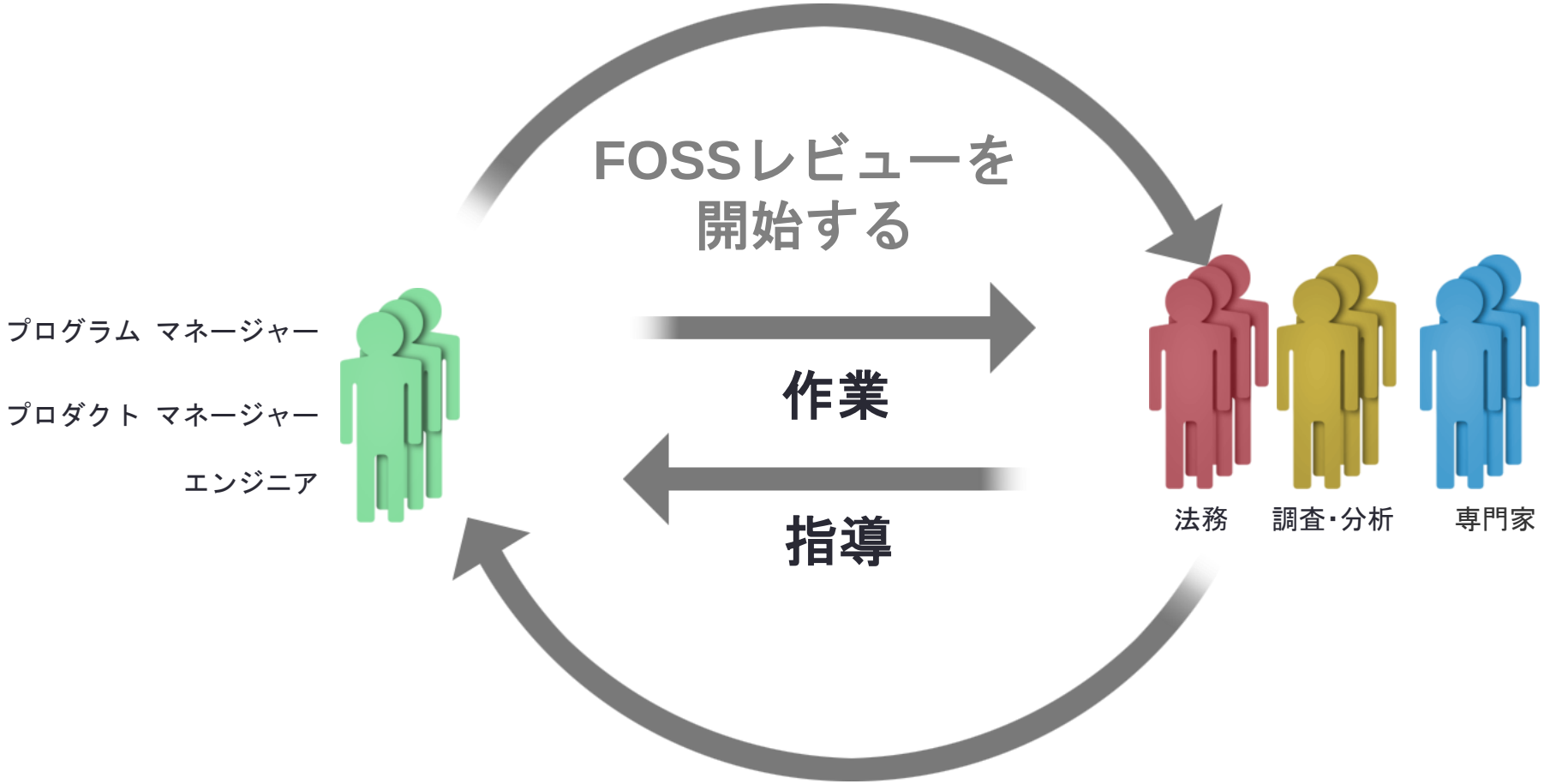
提案されたFOSSの使用を分析する



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。

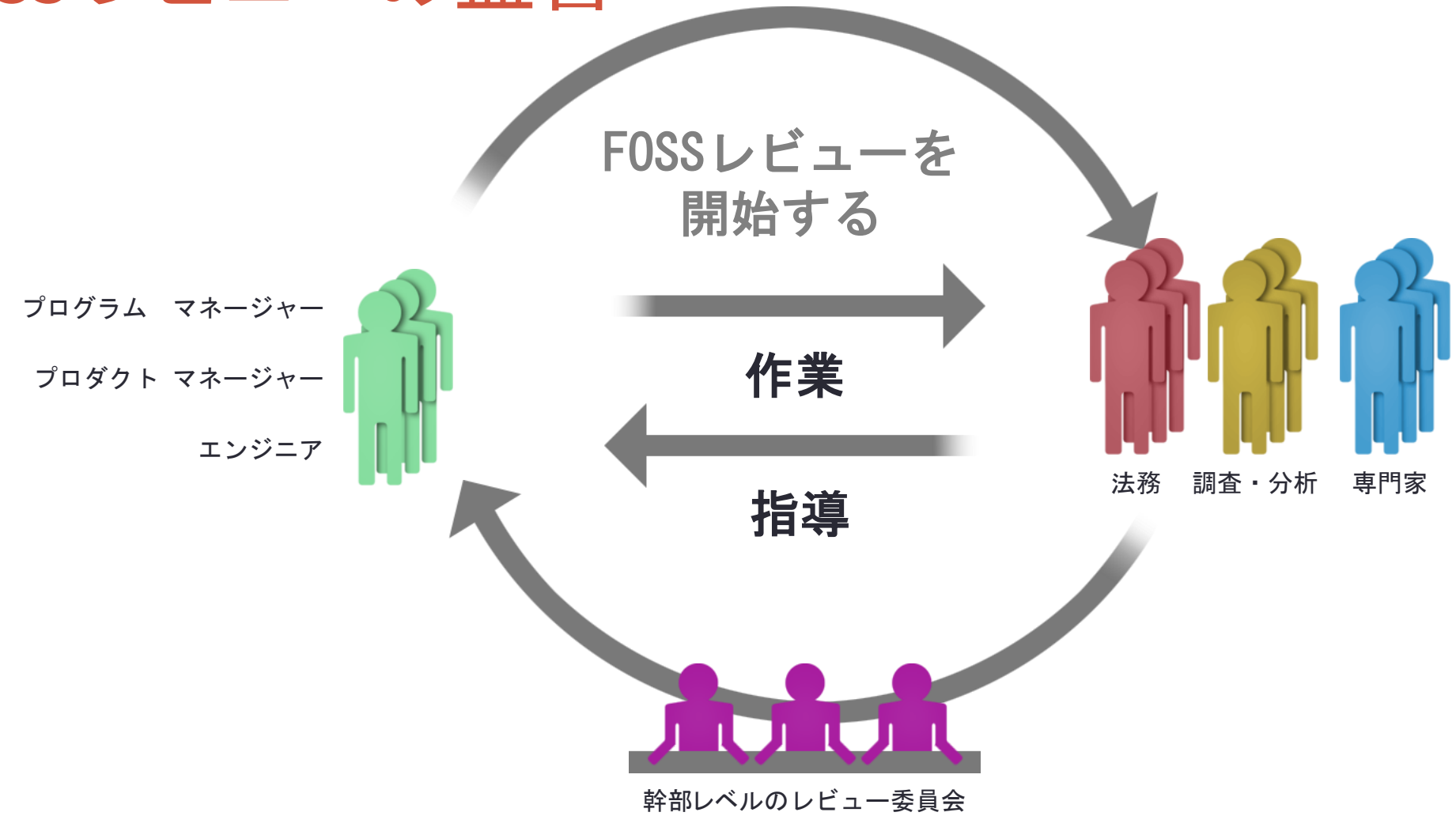
- 完全性、一貫性、正確性（ FOSSの明らかでない使用を精査するためにコード スキャンツールが使われることがある）
- 宣言されたライセンスがコードファイルにある内容と合致しているか？
- そのソフトウェアの使用案を ライセンスが本当に許容しているか？

FOSSレビューの遂行



FOSSレビュープロセスは、インタラクティブに取り組むものとなる。この作業で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ため、フォローアップでの議論では内在する問題をすべての参加者が理解することが求められる。最終的に本プロセスではFOSSの使用についての確実な指導を行う。

FOSSレビューの監督



FOSSレビューのプロセスにおいては、関係者間での意見の相違があったり、ある決定が特別に重要だったりする場合を想定し、十分な監督機能が必要となる。

理解度チェック

OPENCHAIN

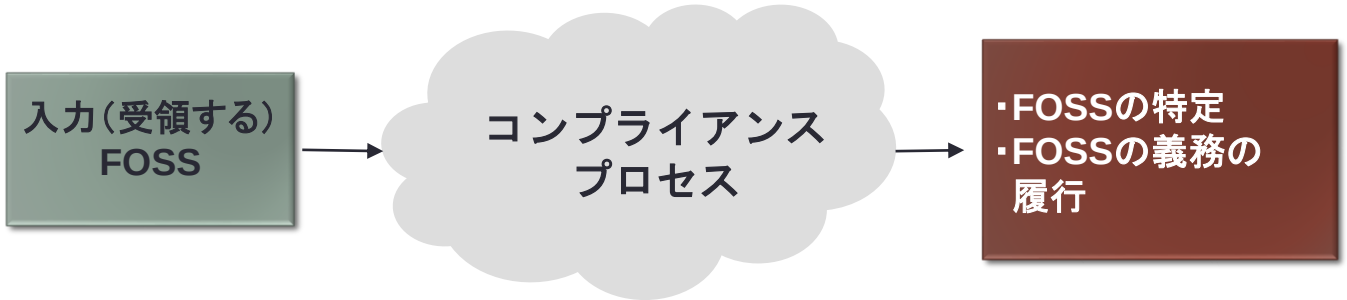
- FOSSレビューの目的は何ですか？
- FOSSコンポーネントを使いたい時に最初に行うべきアクションは何ですか？
- FOSSの使用に関する質問や疑問がある場合、何をするべきですか？
- FOSSレビューのためにどのような種類の情報を集めますか？
- 誰がそのソフトウェアのライセンスを供与しているのかを確認するには、どのような情報が役立ちますか？
- 外部ベンダーから受領したコンポーネントをレビューする際に追加的な情報として重要なものは何ですか？
- FOSSレビューで収集された情報の質を評価するためにどのようなステップを取ることができますか？

第6章

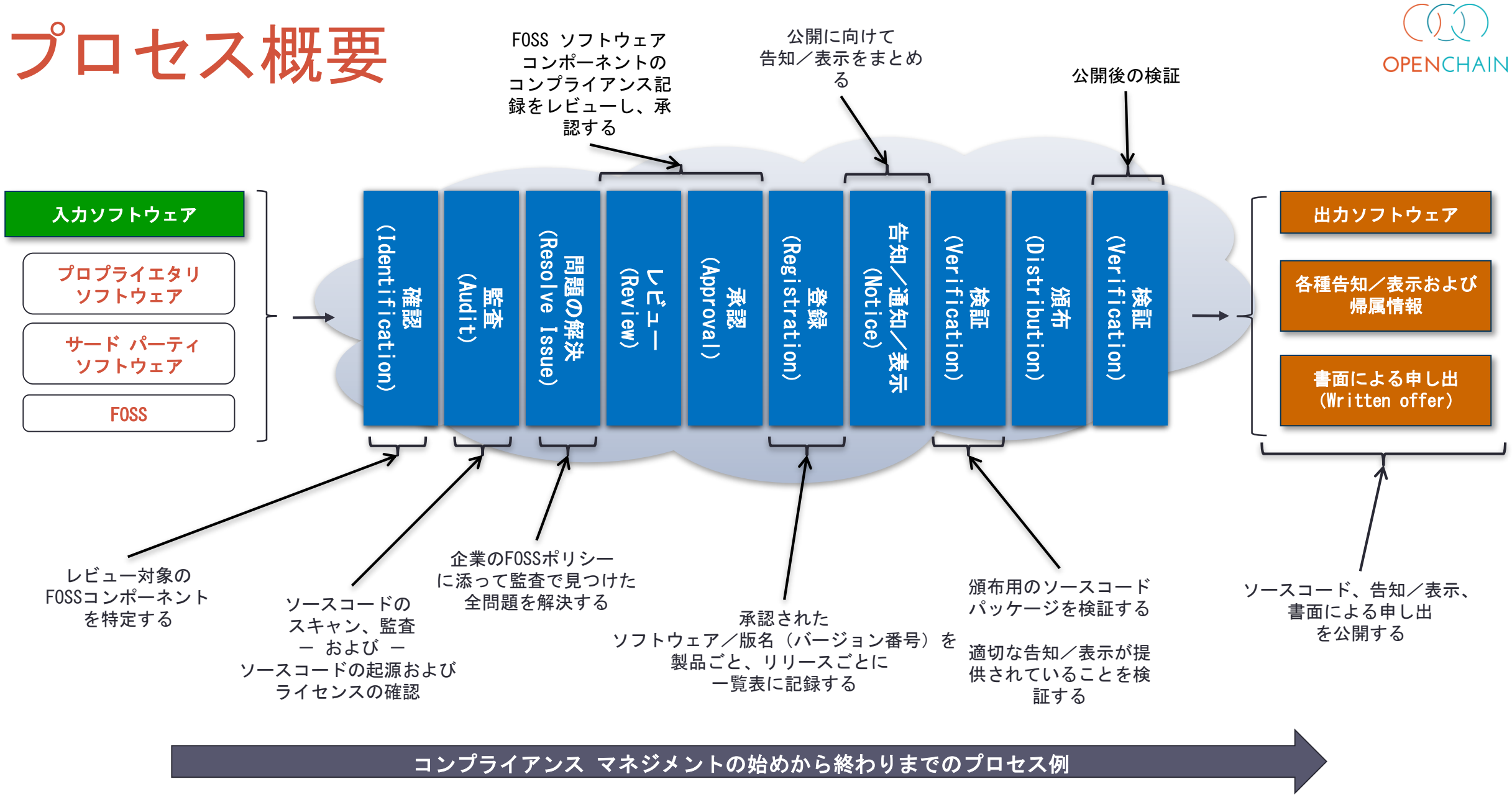
コンプライアンス マネジメントの始めから終わりまで（プロセスの例）

概要

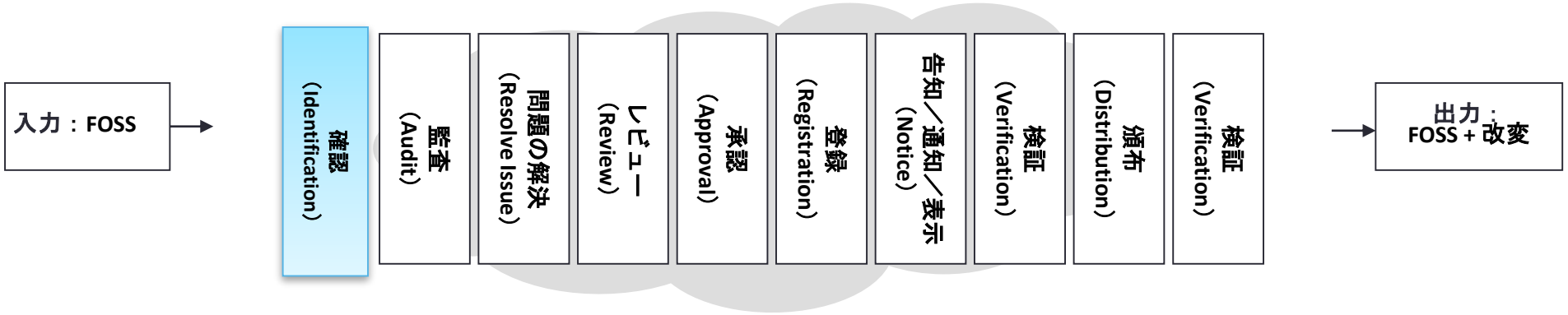
- コンプライアンス マネジメントは、製品 （もしくはOpenChain 仕様書で定義の「供給ソフトウェア」） の中で使われるFOSSの取り込みと頒布をコントロールする一連のアクションで構成される
- コンプライアンスの適正努力（Compliance due diligence）の結果として、供給ソフトウェアで使用されているすべてのFOSSが特定できる。これにより、すべてのFOSSライセンスの義務が履行され、将来にわたり履行されることを確かなものにする
- 大企業が詳細なプロセスを保有する一方で、小規模の企業では単にチェックリストを使うだけの場合がある。本章では大企業のプロセスの一例を紹介する



プロセス概要


FOSS ソフトウェア コンポーネントの コンプライアンス記 録をレビューし、承 認する
公開に向けて 告知／表示をまとめ る
公開後の検証
入力ソフトウェア
プロプライエタリ ソフトウェア
サード パーティ ソフトウェア
FOSS
確認 (Identification)
監査 (Audit)
問題の解決 (Resolve Issue)
レビュー (Review)
承認 (Approval)
登録 (Registration)
告知／通知／表示 (Notice)
検証 (Verification)
頒布 (Distribution)
検証 (Verification)
出力ソフトウェア
各種告知／表示および 帰属情報
書面による申し出 (Written offer)
レビュー対象の FOSSコンポーネント を特定する
ソースコードの スキャン、監査 ー および ー ソースコードの起源および ライセンスの確認
企業のFOSSポリシー に添って監査で見つけた 全問題を解決する
承認された ソフトウェア／版名（バージョン番号）を 製品ごと、リリースごとに 一覧表に記録する
頒布用のソースコード パッケージを検証する
適切な告知／表示が提 供されていることを検 証する
ソースコード、告知／表示、 書面による申し出 を公開する
コンプライアンス マネジメントの始めから終わりまでのプロセス例

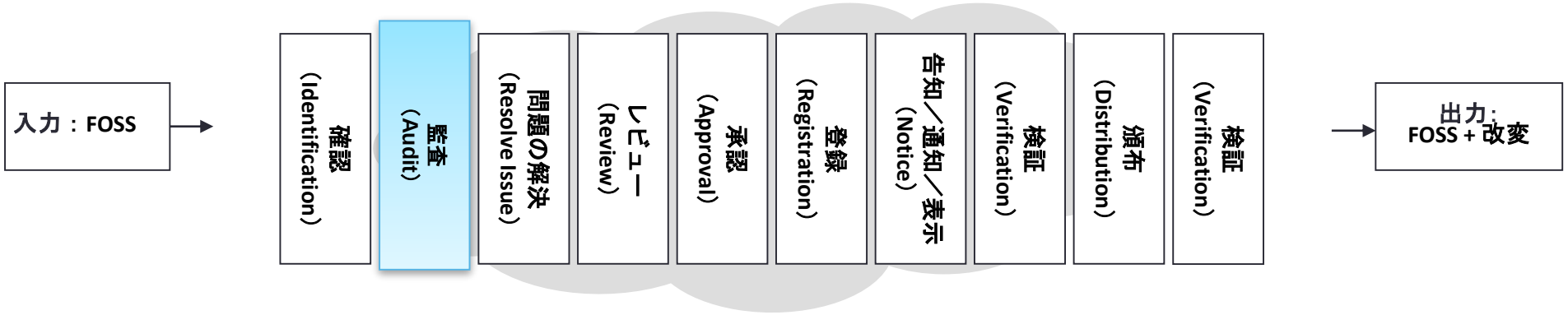
FOSSの使用を確認し、追跡する



すべてのソースに含まれるFOSSを確認し、追跡を開始する

- 前提条件：
  - このプロセスは以下のイベントのうちの1つで開始される：
    - 開発チームがFOSSコンポーネントのレビューや外部向けのリリースを要望する
    - 適切な承認がなく使用されているFOSSを発見する
    - サード パーティのソフトウェアの一部に使用されているFOSSを発見する
- ステップ：
  - 入力リクエストが登録される
  - 全プラットフォームのスキャンが実施される
  - サードパーティ提供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ているが、入力リクエストのないすべてのFOSSコンポーネントを識別し、レビューを実施する
- 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
  - ソースコードのスキャンまたはレビューのための（次のステップとなる）監査が要請される

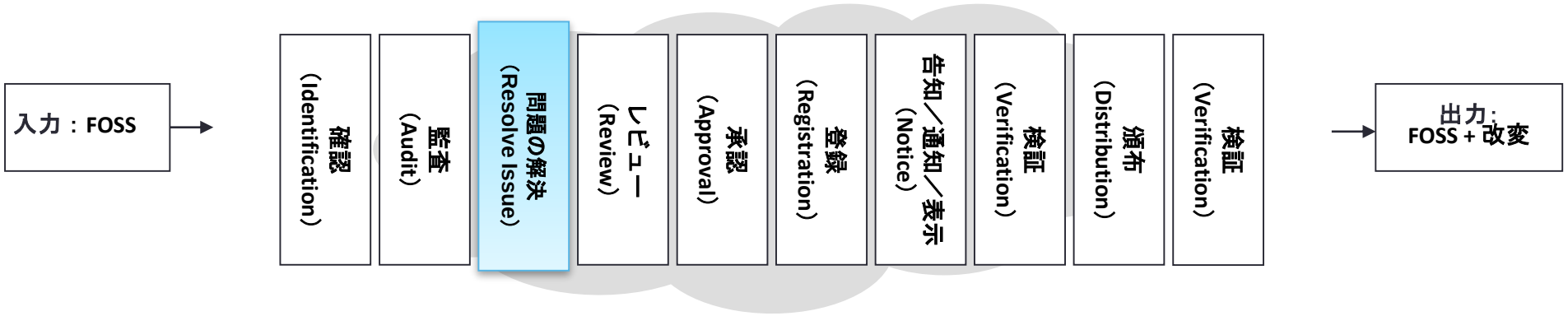
ソースコードを監査する



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

- 前提条件：
  - 開発チームがコンプライアンスの記録をFOSSの使用方法に関する情報と併せ提供する
  - 開発チームから提供される記録がない場合、FOSSコンポーネント発見時に記録が生成される
- 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：
  - 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生成される

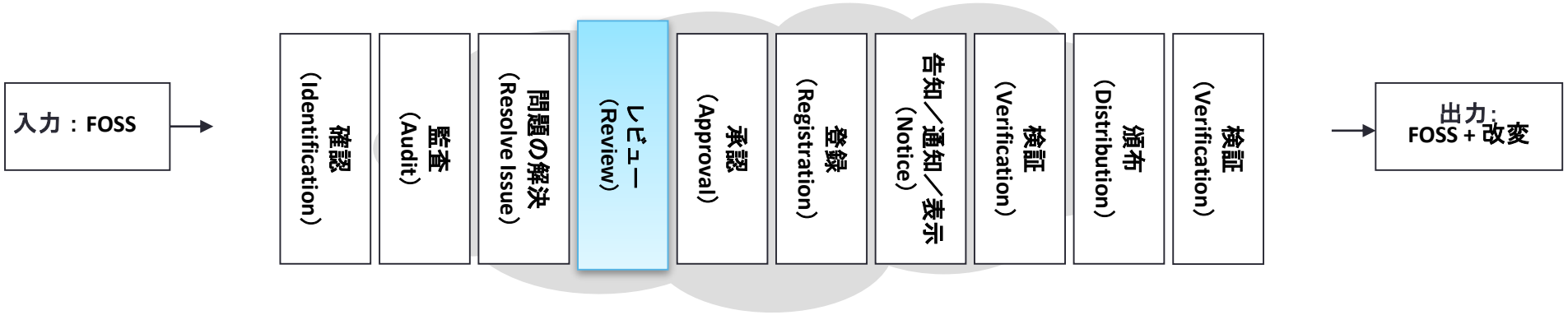
問題を解決する



監査で確認されたすべての問題を解決する

- 前提条件：
  - ソースコードの監査やスキャンが完了している
  - 監査レポートがソースコードの起源とライセンスを特定し、さらなる調査が必要なファイルにフラグが立てられている
- 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
  - 問題が解決されたことをエンジニアとともに確認する
- 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

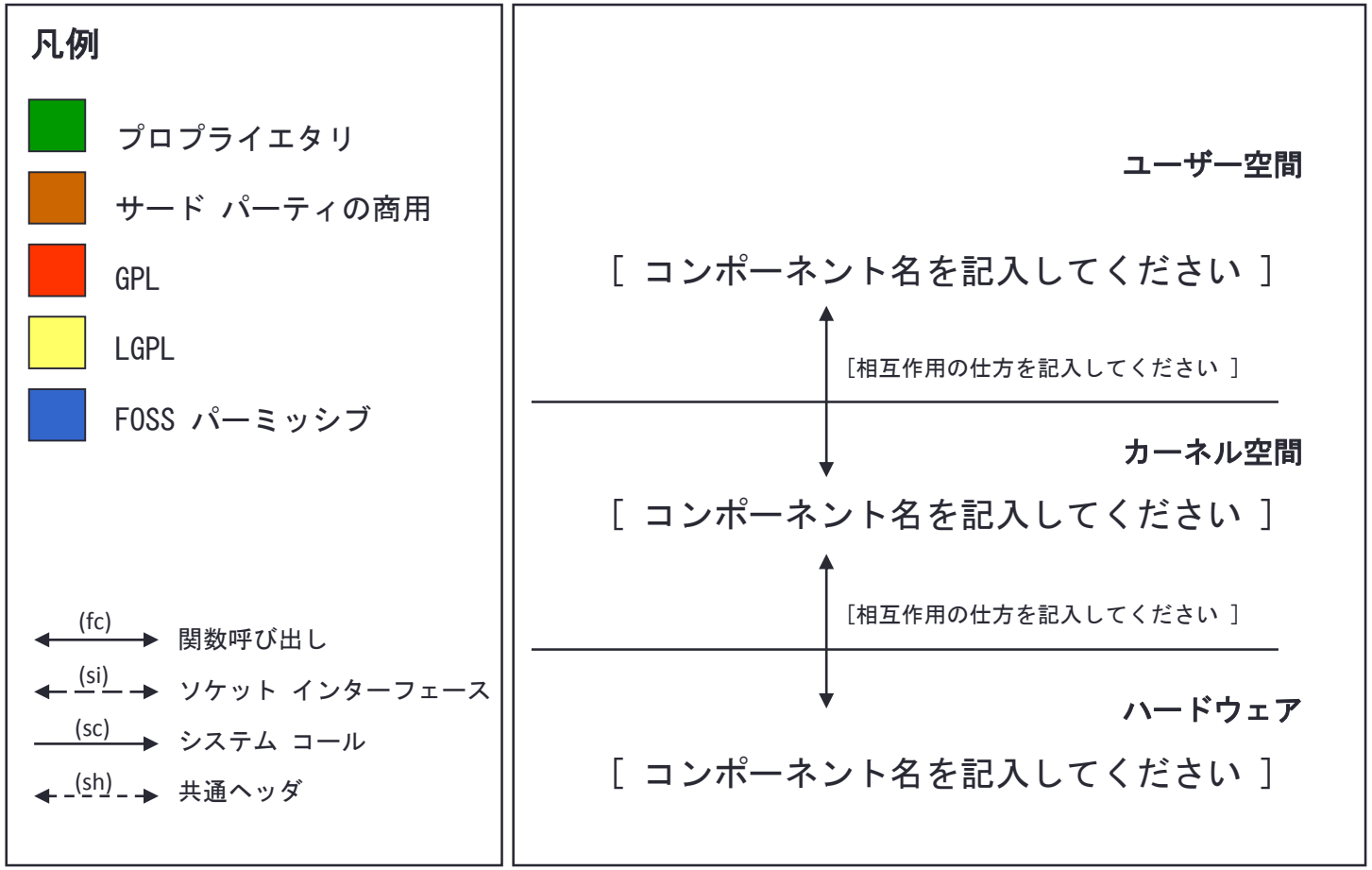
レビューを実施する



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

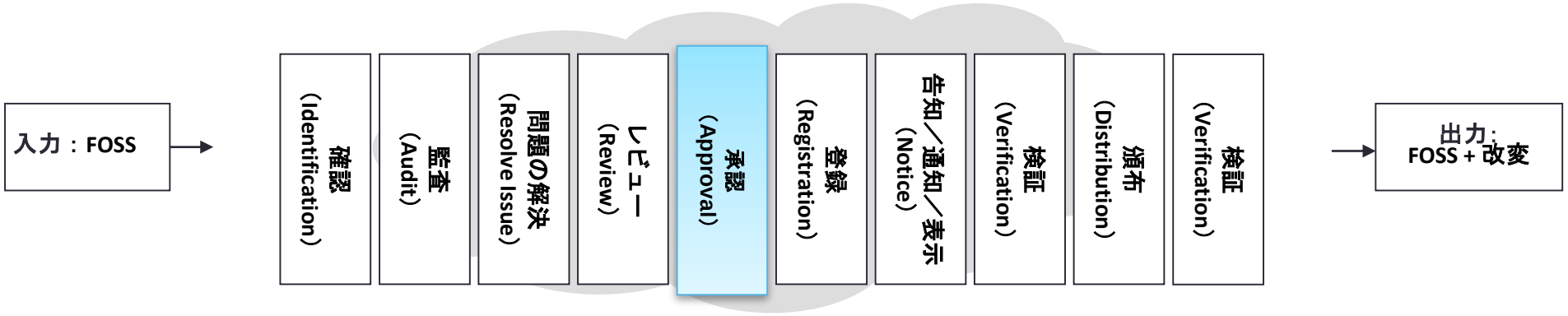
- 前提条件：
  - ソースコードが監査されている
  - すべての指摘された問題が解決されている
- ステップ：
  - レビュー スタッフに適切な職権レベルを含める
  - 監査されたソースコード、ソフトウェア アーキテクチャ、およびFOSSの利用方法についてFOSSレビューを実施する （次スライドのテンプレート参照）
  - FOSSライセンス下の義務を確認する
- 成果：
  - 監査レポートにあるソフトウェアがFOSSポリシーと合致することを確かなものとする
  - 監査レポートで発見されたことを保存し、解決された問題を次のステップへの準備ができた（つまり承認された） ものとして示される

アーキテクチャ　レビュー（テンプレートの例）



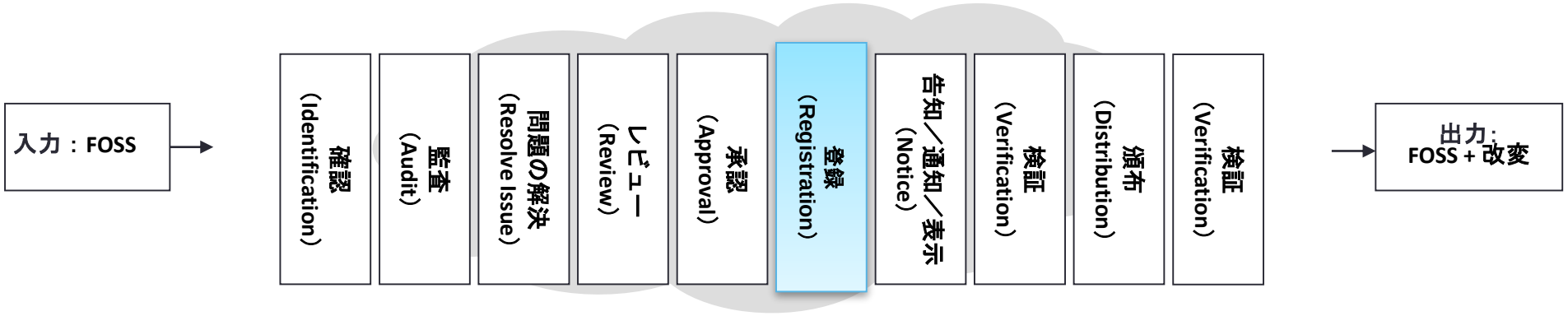
承認

- 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

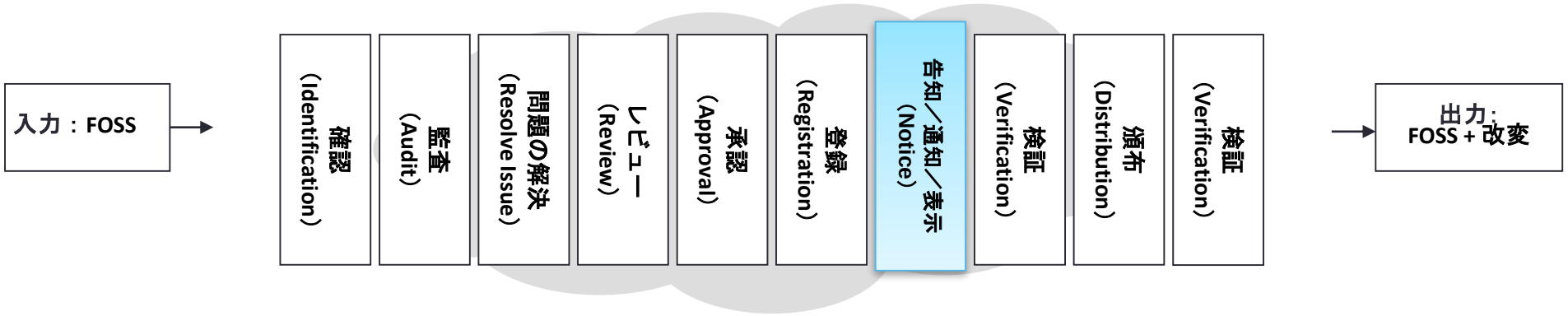


登録／承認の追跡

- 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



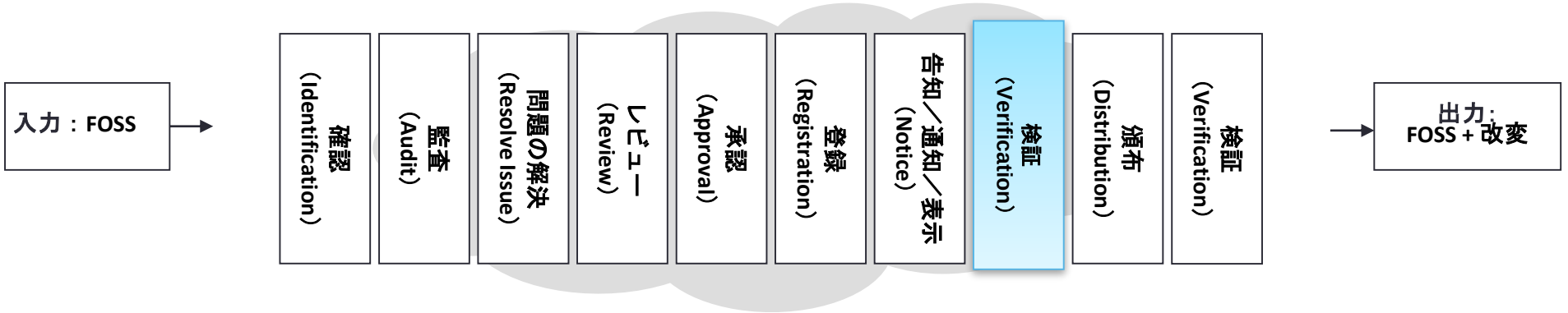
告知／通知／表示



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

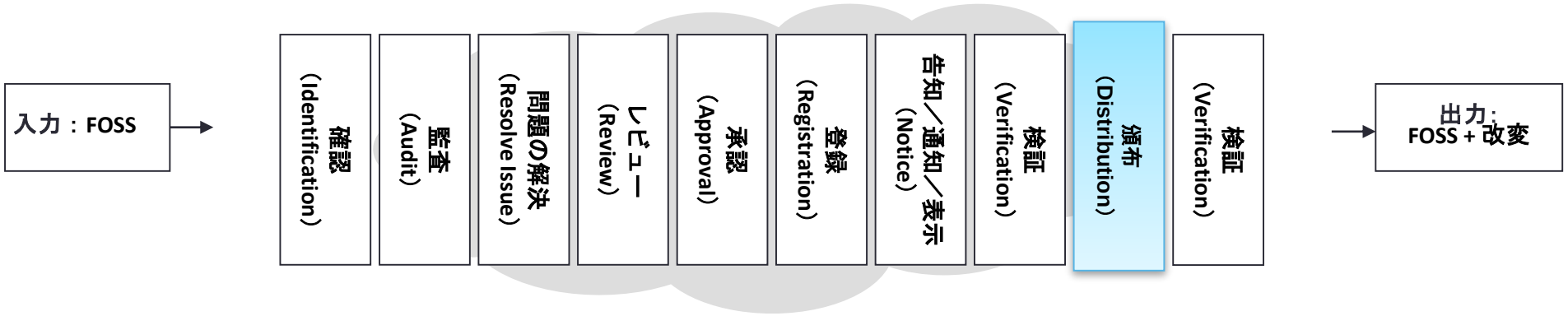
頒布前の検証



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

- 前提条件：
  - FOSSコンポーネントの使用が承認されている
  - FOSSコンポーネントがそのリリースのソフトウェア一覧表に登録されている
  - 適切な告知／表示が準備されたている
- 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

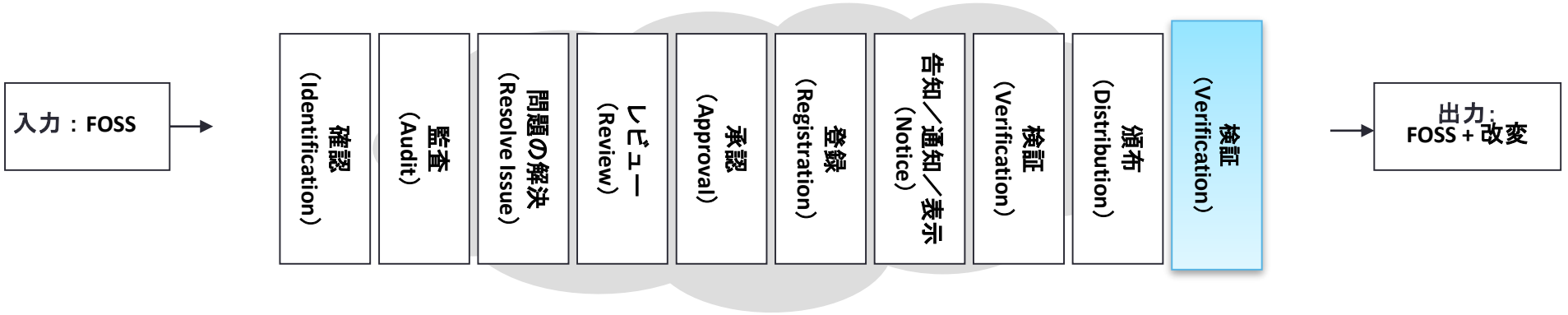


添付ソースコードを要求された形で提供する

- 前提条件：
  - すべての頒布前検証が完了し、問題が発見されていない
- 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

- 前提条件：
  - 添付ソースコードが要求された通りに提供されている
  - 適切な告知文が準備された
- 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 レビューではど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
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ポリシーに反して、（自社にとって、もしくはサード パーティにとって）プロプライエタリなソースのコードにFOSSコードを追加（またはカット＆ペースト）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： ・ コンプライアンスでの問題、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意識されるように、技術スタッフにトレーニングを提供する ・ ビルド環境においてすべてのソースコード（プロプライエタリ、サード パーティ、FOSS）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

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なソフトウェアに意図せずにリンクされてしまう（逆もまた同様） 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（FOSS、プロプライエタリ、サード パーティ） 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

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
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	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

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 。

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。

製品出荷前にコンプライアンスを確認する

OPENCHAIN

- 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？