TOSHIBA

オープンソースカンファレンス 2026 Tokyo/Spring

製品開発における実務的なOSS管理の基本

2026.2.27
株式会社東芝 総合研究所
デジタルイノベーション技術センター OSS推進部
武山 文信

本日の講師について

- 武山 文信
  - 趣味の時間で OSC などでコミュニティ活動をしていたら、
    OSS の部署に連れていかれた
- OSPO (Open Source Program Office)
  - OSS の活用、開発、**ルール作り**、貢献・コミュニティ連携などを横断的に管理する組織
  - 国内外の各企業が OSPO を設置、最近は国内でも注目が高まる
    - トヨタ・日立が挑む無償ソフトへの貢献「デジタル下手な国」返上を （日本経済新聞 25/9/8）
    - 特集先行企業に学ぶ、OSPOの勘所（日経XTECH, 日経コンピュータ 25/4/3）
    - OSPO Landscape

**皆さんの所属企業や大学には
OSPO はありますか？**

東芝グループの事業の全体イメージ

OSS は高信頼・高付加価値の製品を速やかに届けるために不可欠

皆さんへメッセージ

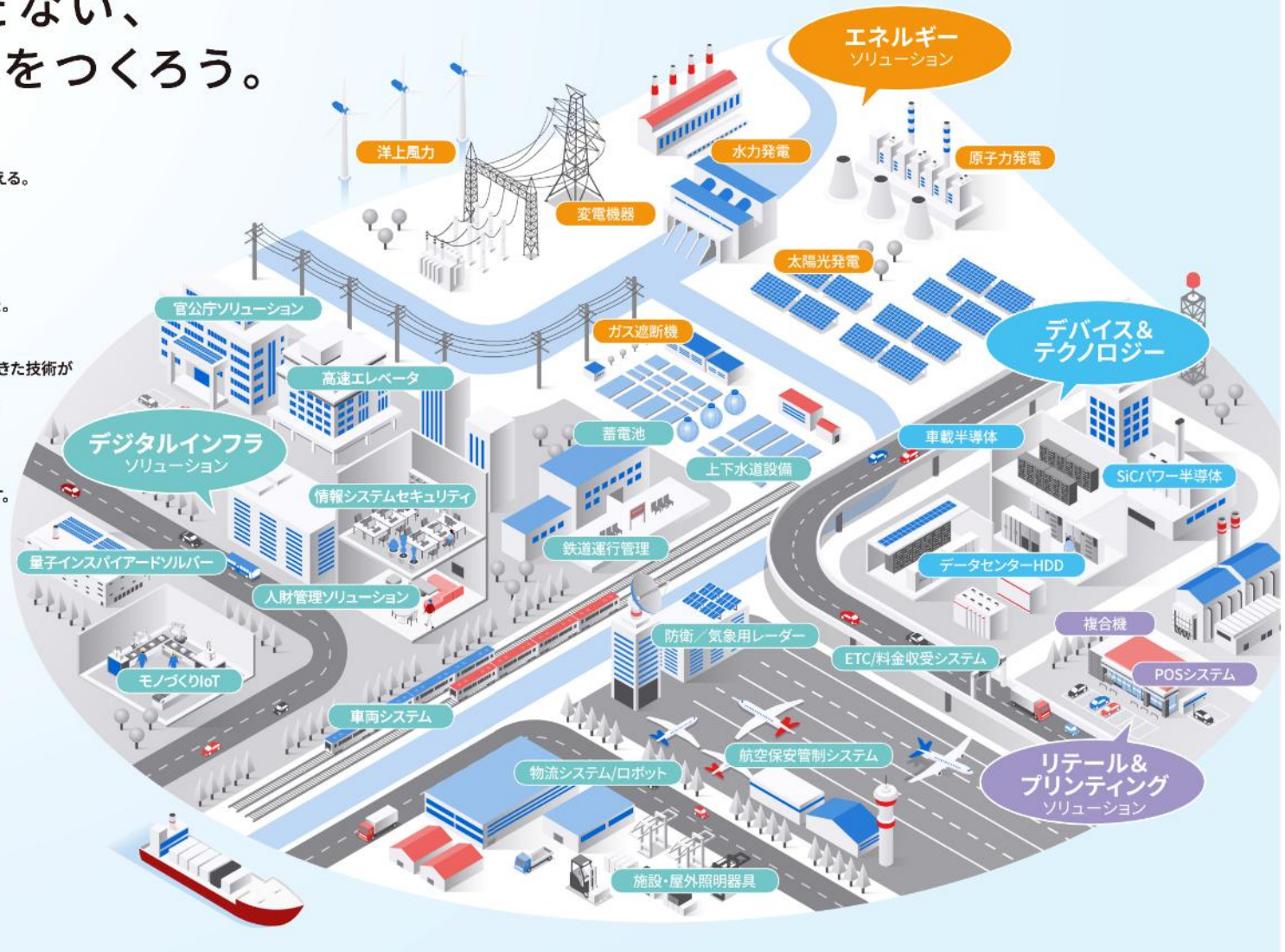
世界にまだない、あたりまえをつくろう。

24時間きれいな水が飲める。
停電を心配することなく電気が使える。
時間通りに電車がくる。

そんな「あたりまえ」を、
東芝はものづくりで支えてきました。
東芝の先人たちが、
多くの困難を乗り越えて生み出してきた技術が
「今のあたりまえ」になっています。

ものづくりを超えて、
私たちは、これからも挑み続けます。
「世界にまだない、あたりまえ」を
つくることに。

ともに、世界を前に進めよう。
人と、地球の、明日のために。



TOSHIBA RECRUITMENT GUIDE より

参加者のみなさんへ質問

みなさんは、どのように OSS を使用していますか？

1. **自社製品、自社サービスの開発で OSS を利用している**
2. **業務を請け負って製品・サービスを開発する際に OSS を利用している**
3. **OSS を使用しているが、開発はしていない**

本日は
こちらのみなさんを主な対象に
OSS管理について話します

01

OSS管理と大変さ

製品やサービスに含まれるOSSを管理する必要がある理由

ライセンスコンプライアンス

- OSS はライセンスで定められた条件の下、自由に利用、配布、改変ができる
  - 条件の例: 著作権表示、ライセンス文の表示・提供、ソースコードの開示・提供
- すべての OSS を把握しておかないと、条件を満たすために必要なことができず、ライセンス違反につながる可能性

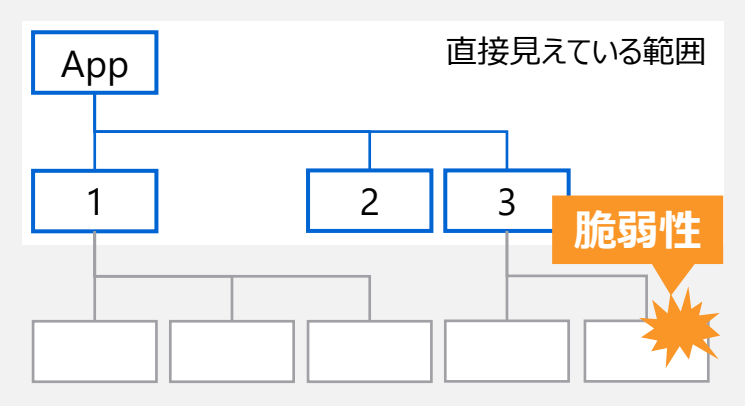
セキュリティアシュアランス/脆弱性管理

- 製品に含まれる脆弱な OSS を見つけ、その脆弱性が製品に影響がないか判断し、必要に応じてバージョンアップ等の対策を行う必要がある
- すべての OSS を脆弱性管理システム等に登録しておく必要がある

使用している OSS とは？

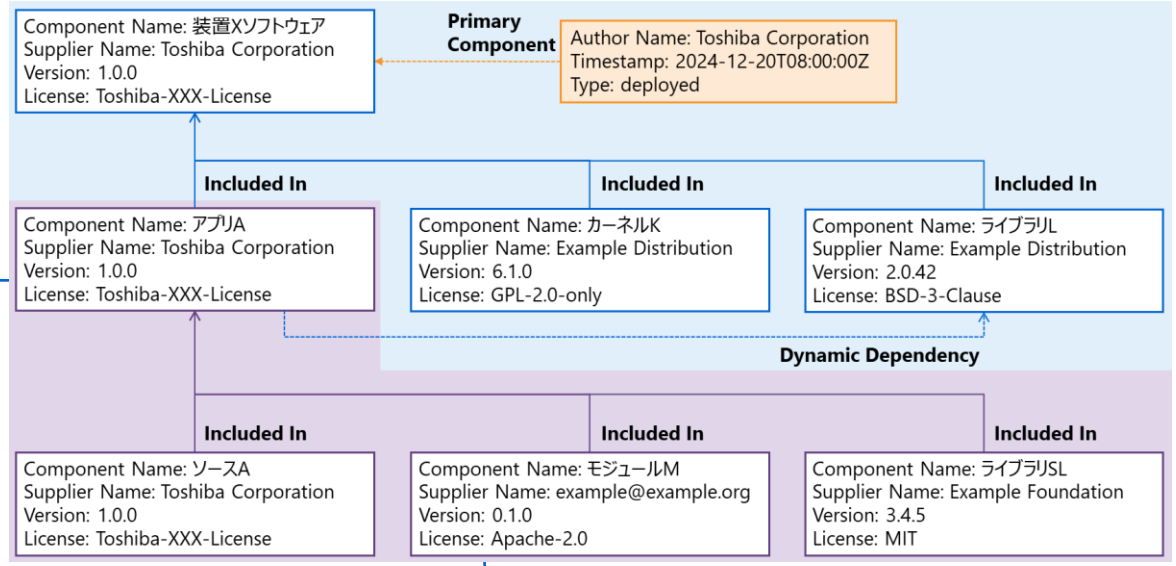
- 直接使用している OSS に加え、
  依存関係としてインストールされた OSS も管理しなければならない
  - 例えば:
    `$ pip install pandas`
    ➡ numpy, python-dateutil, six も依存関係としてインストールされる
  - 依存関係としてインストールされたライブラリが重大な脆弱性になることも
    - Apache Log4j 2 の任意コード実行の脆弱性 (Log4Shell CVE-2021-44228)
      - ログメッセージに出力される文字列を悪用して、リモートから任意コードの実行が可能

- 使用している OSS の数は多くなりがちで管理が大変
  - アプリケーションで使用するライブラリ: 100～
  - Linux ディストリビューションのパッケージ: 150～



SBOM (Software Bill of Materials) 1/2

- ソフトウェアを構成する OSS をはじめとした**コンポーネントの一覧**と、
  そのコンポーネントのサプライチェーンに関わる**形式的（機械可読）な記録**
  - コンポーネント情報: 名前、バージョン、ライセンス、…
  - コンポーネント間の関係
- SBOM 文書
  - SBOM を SPDX や CycloneDX などの
    標準化されたフォーマットで
    出力したもの




```
{
  "SPDXID": "SPDXRef-Package-rpm-PyYAML-c59e1762d203a699",
  "copyrightText": "NOASSERTION",
  "downloadLocation": "NOASSERTION",
  "filesAnalyzed": false,
  "licenseConcluded": "NOASSERTION",
  "licenseDeclared": "MIT",
  "name": "PyYAML",
  "originator": "Organization: Oracle America",
  "sourceInfo": "acquired package info from RPM DB: var/lib/rpm/Packages",
  "supplier": "Organization: Oracle America",
```

SBOM (Software Bill of Materials) 2/2

- 世の中的に SBOM が求められるようになっている
  - 製品を顧客への納品時に SBOM 文書を提出
    - 顧客が製品に含まれる OSS を把握するために必要
    - 最終製品だけではなく、ソフトウェア外注時も
  - 法令、認証への準拠
    - 欧州サイバーレジリエンス法: 製造者の義務として SBOM の作成
    - JC-STAR（今後）

SBOM を提供するためには、OSS を適切に管理できていないといけない

このようなことは起きていませんか？

- 製品が完成した後に使用している OSS を調べたら
  - ライセンス上利用できないものが含まれていて、差し替えが必要になった
  - 重大な脆弱性があり、バージョンアップが必要になった
- 製品内にどのような OSS が含まれているか分からなくなっている
  - ソースツリー内に OSS っぽいファイルがある
  - OSS っぽいコピペコードがある

02

OSS 管理の方法

おことわり

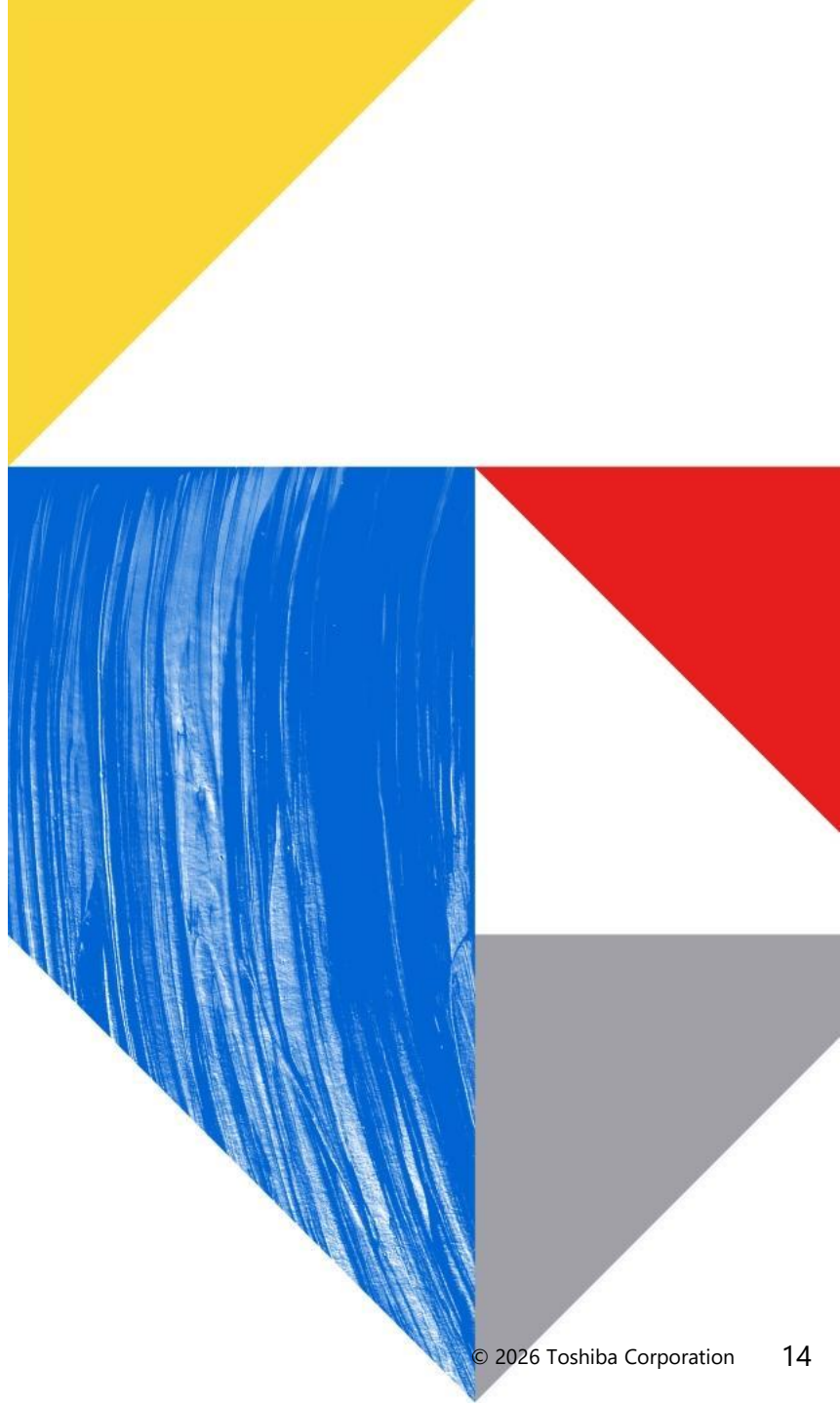
- OSS 管理について、現時点で絶対的な方法論や教科書があるわけではありません
  - 製品の特性によって最適な方法も異なります
- このセミナーはパブリックベータ（アルファ？）のような位置づけです
  - 参加者のみなさんからの質問や、経験の共有も歓迎です
  - 「教科書」を共創していければと思います

OSS 管理のポイント

- OSS 管理の**方針（ポリシー）**をあらかじめ定め、
  必要な準備をしてから開発をすることが大切
  - **A: OSS の情報と成果物の管理方法**
    - 成果物: アーティファクト、実体となるファイル
  - **B: 利用可能なライセンス**
    - 現在の開発中の製品で使用して良いライセンスは何か？
  - **C: OSS を管理する体制・プロセス**
    - いつ、誰が OSS を利用状況を確認するか？
- 定めた方針は**文書化**して関係者と共有する
  - まずは、社内 Wiki に書く・・・といった方法でもよい
  - 方針のレビュー方法や周知の方法も整備できるとベスト

A

OSSの情報と成果物の管理方法



決めなければいけないこと

1. OSS の情報を記録する方法
2. OSS の成果物を保管する方法（実体となるファイルなど）を
   自社のコード、バイナリ等の成果物と区別して保管する方法
3. OSS の特定方法と範囲
4. 脆弱性管理方法

OSS の管理しておくべき情報

項目	説明
名前	OSS の開発元が決めた、この OSS の名前
提供元	OSSの提供者。ディストリビューションの一部として提供されている場合、オリジナルの開発者と提供元が異なる場合がある
バージョン	どのリリースであるかを特定する文字列
ライセンス情報	この OSS がどのライセンスで利用できるか
入手元	オンラインで入手できる場合、ファイルを入手できる URL
依存関係	このOSSを使用するために必要な他のコンポーネント。バージョンアップやパッケージを削除する際に必要。
脆弱性管理に必要な識別子 (ID)	脆弱性データベースから検索する際にキーとして使用できる識別子。

パッケージマネージャー

みなさん、ご存知

OSS の情報を記録する方法
OSS の成果物を保管する方法

- 前頁の情報を管理してくれる便利ツール
  - 言語系: pip (Python), npm (NodeJS), gradle (Java), cargo (Rust), …
  - Linuxディストリビューション: dpkg, rpm
- 成果物の管理もしてくれる

- パッケージマネージャーを原則使用する方針を定めておくとよい
  - 方針を定めておかないと:
    Web 用のアイコンやフォントを npm ではなく、zip ファイルをダウンロードしてソースツリーに保存してしまった（バージョン等が分からなくなってしまう）

パッケージマネージャーを利用できない場合

OSS の情報を記録する方法
OSS の成果物を保管する方法

- OSS の情報を手動管理するしかないが、決定的な方法が無いため、
  開発時にどうするか決めておく必要がある
- OSS の情報を残す方法
  - Excel でドキュメントとして管理する
  - ソースコード内に OSS の情報を記録したファイルを置く
    - REUSE.toml （**次ページ**）
    - Google Third Party METADATA
    - SBOM 文書（SPDX, CycloneDX）
  - ・・・
- 自社のコード/バイナリと OSS が混ざらないように成果物を保管する方法
  - ソースツリーの third_party ディレクトリ内に OSS を保管する
    - Git サブモジュールを利用すると、実ファイルをリポジトリから分離し、入手元もはっきりするのでよい
  - バイナリの場合: /opt/... 以下にインストールする

実施例: REUSE.toml

- ソフトウェアプロジェクトの著作権とライセンス情報を宣言するためのファイル
  - ソースツリーに REUSE.toml を置き、ファイルごとに情報を付与することができる
    - 著作権とライセンス情報以外の情報は、
      SPDX のタグが使えることにはなっているが、標準化されていないのが難点

```
version = 1

[[annotations]]
path = ["third_party/libFoo/**"]
SPDX-FileCopyrightText = "2026 Foo Foundation"
SPDX-License-Identifier = "MIT"
SPDX-PackageDownloadLocation = "http://foo.example.org/download/foo-1.0.0.tar.xz"
SPDX-PackageVersion = "1.0.0"

[[annotations]]
path = …
  :
```

ファイルパターンを指定して
OSS情報を割り当て

標準化されて
いないので注意

OSS の特定方法と範囲

- 特定: OSSの「管理するべき情報」を明らかにする
  - パッケージマネージャーあり
    - **理想的には**パッケージマネージャに問い合わせるだけでよい
  - パッケージマネージャーなし
    - 入手時に名前やバージョン、入手元は記録するだけであるが、ライセンスは調査が必要になる場合がある
  - 入手時にOSSの情報が管理されていないソースコード

問題があった場合の影響の大きさ、パッケージの種類によって決める

- 特殊ケースへの対応: どこまで特定を行うか、範囲を決める必要がある
  - パッケージの内部に含まれている別の OSS も特定するか？
    - opencv-python の事例（次頁）
  - パッケージマネージャのライセンス情報の妥当性をどこまで確認するか？
    - Fedora の libzstd のパッケージのライセンスが間違っていた事例

opencv-python の事例

- opencv-python
  - Python で画像・動画処理の定番ライブラリの OpenCV を扱うパッケージ
  - パッケージメタデータ上のライセンス: **Apache-2.0**
- 課題: pip で取得するパッケージ内に多数のライブラリが同梱されている
  - Qt: LGPL-3
  - FFmpeg: LGPL-2.1
  - ・・・

Apache-2.0 以外のライセンス

脆弱性は？

```
libQt5Core-195a14c9.so.5.15.18
libQt5Gui-26aef175.so.5.15.18
:
libavcodec-156beeea.so.62.11.100
libavdevice-13b3e407.so.62.1.100
libavfilter-93b41299.so.11.4.100
libavformat-8c8a026e.so.62.3.100
:
libxkbcommon-x11-27b61ed0.so.0.0.0
```

ドキュメントには、同梱されるライブラリの情報が記載さているが、
パッケージ情報だけでは分からない
パッケージが信頼できないというより、あまり好ましくない特殊ケース

特殊ケース・適切に管理されていない場合に使えるツール

- ライセンス・著作権情報スキャナー
  - ソースコード中に含まれるライセンスや著作権関連の文字列を見つける
  - 例: scancode, Fossology, …
- 大規模データベースを使用したスキャナー
  - OSS のファイルを大量に収集したデータベースを使用し、ファイルの一部やコードの一部（スニペット）が OSS の一部であるかを判定
  - 例: BlackDuck, ScanOSS, …
- ツールを使用する場合はあらかじめ準備をしておく

強力なツールであるが、見落とし、誤検知、過検知があるため過信は禁物

脆弱性管理方法 1/2

- 大量のパッケージの脆弱性情報を毎日手動で検索はできないため
  脆弱性管理ツールを活用することが大切
- 脆弱性管理ツールの役割
  - 脆弱性データベースの定期チェック
  - ステータスの管理
    - 影響の調査中
    - 影響なし
    - 影響あり対策中
    - 対策完了
- ツールの例
  - FutureVuls, Dependency Track, Vul Hummer, …
  - Trivy, Syft & Grype, …

脆弱性管理方法 2/2

- 脆弱性情報のマッチングに**使用する情報が変わる**ため、**ツールを先に決めておく**
  - CPE (Common Platform Enumeration)

```
cpe:2.3:a:haxx:curl:8.14.1:*:*:*:*:*:*:*
```

    - NVD (National Vulnerability Database) から脆弱性を検索するベーシックな方法
    - 課題: 同じ OSS の配布元違いを区別できない場合がある
    - 課題: 脆弱性が見つかるまで ID が決まらないのが難点
  - PURL (Package URL)

```
pkg:deb/debian/curl@8.14.1-2+deb13u2?distro=trixie
```

    - パッケージ情報をもとに決められた検索用 ID
    - パッケージ提供元の脆弱性情報を使用した、**きめ細やかな脆弱性管理**
    - 脆弱性DBに PURL を登録する動き
    - 課題: すべての PURL が脆弱性管理に対応しているわけではない

OSS の入手場所にも影響
PURL による脆弱性情報の検索に対応した場所から OSS を入手する必要あり

B

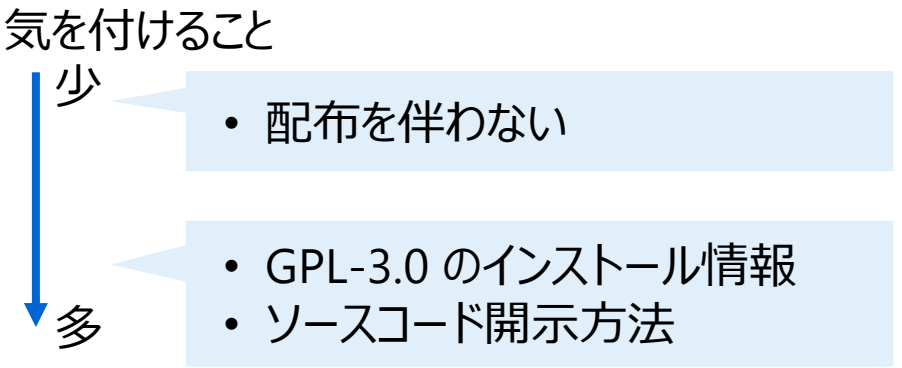
利用可能なライセンスの管理

OSS ライセンス

- OSS ライセンスは無限にあるわけではない
  - 新しいライセンスを作らないことがベストプラクティス
- OSS ライセンスを参照するときは **SPDX License Identifier** を使用するとよい
  - ライセンスに短い ID を割り当て
    - BSD 3-Clause “New” or “Revised” License ➡ BSD-3-Clause
    - Apache License 2.0 ➡ Apache-2.0
  - 695 個の ID が登録されている（Ver. 3.28.0）
    - 頻繁に使用するのはごく一部

利用可能なライセンス

- 利用可能なライセンスをあらかじめ決めておくと、スムーズに判断ができる
  - 開発中に OSS を選定するとき、ライセンス上利用可能かを判断する場面は多い
  - 未決定のライセンスに出会った場合は、利用可能かを判断して追加する
- 製品の特性により変わる
  - 自社管理のサーバー/クラウド上で運用するサービス
  - B2B 製品
  - コンシューマー製品



利用可能ライセンスの管理方法

- 利用用途ごとに決める必要がある
  - アプリケーションにリンク・組み込んで使う
    - 製品のソースコードの開示不可なら
      パーミッシブライセンス (MIT, BSD-3-Clause, Apache-2.0, ...)、準コピーレフトライセンス (LGPL-2.1 など) を許可
  - 実行環境にインストールして使う
    - 製品の特性次第で、コピーレフトライセンスも利用可能
- ライセンス本文を確認して問題ないかを判断した結果をデータベース化しておくとよい
  - 知財、法務部門の有識者と行うことを推奨
  - いずれの用途でも利用不可とした場合も理由を記録

管理方法の例

SPDX Identifier	種類	アプリ	実行環境	備考
MIT	パーミッシブ	可	可	
Apache-2.0	パーミッシブ	可	可	
LGPL-2.1-or-later	準コピーレフト	可	可	
GPL-3.0-or-later	コピーレフト	不可	可	
XXXX-XX	-	不可	不可	XXXの条件があいまいであり、満たすことが困難なため、利用不可

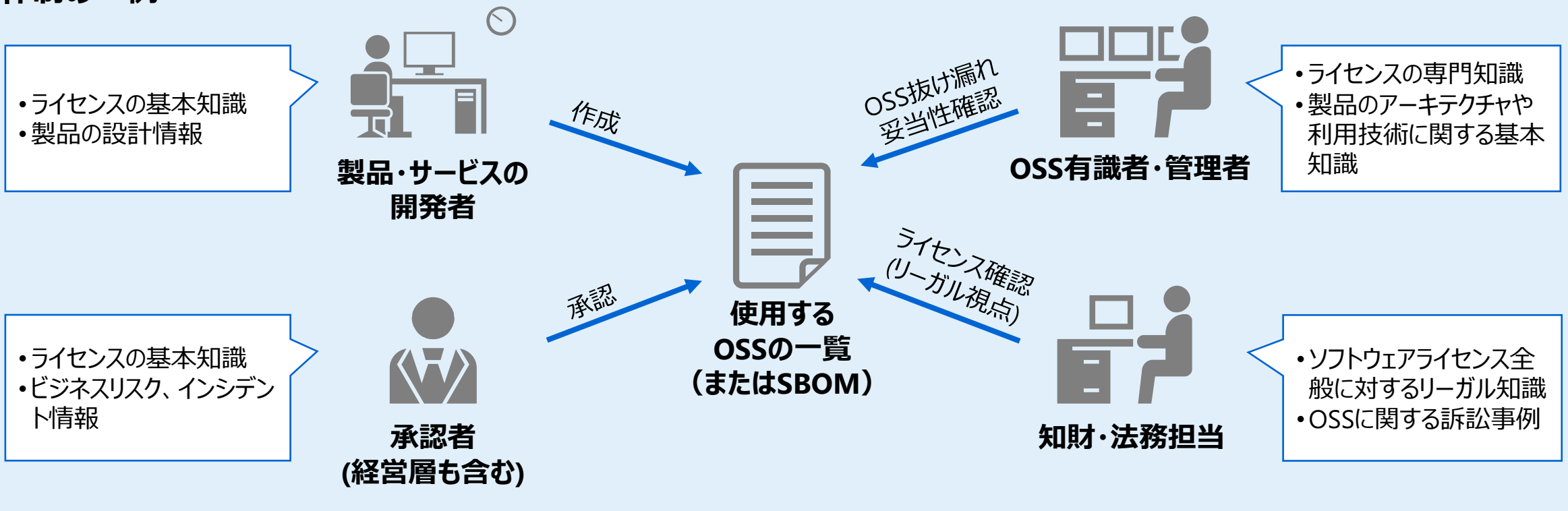
C

体制・プロセス

体制と力量定義

- これまで管理してきた OSS の一覧をもとに、製品への適用可否を**組織で判断**する
- 役割と力量（必要なスキル）を決め、実際に業務を行う体制を構築する

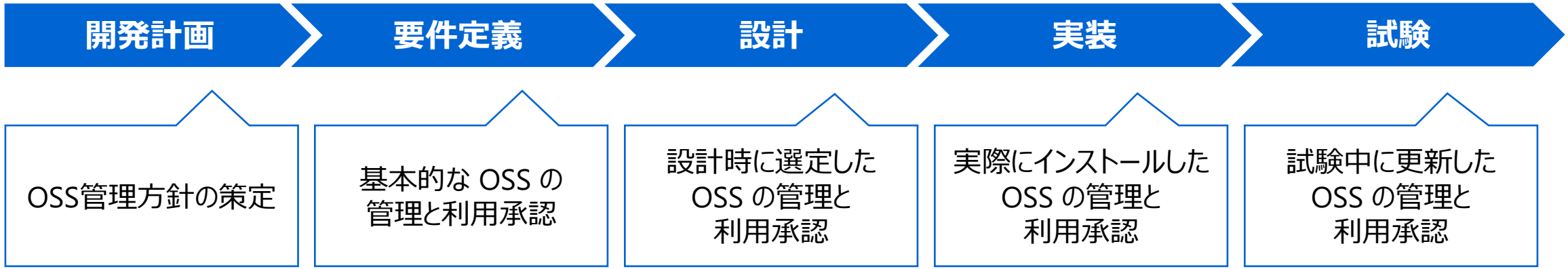
体制の一例



プロセス

- どのタイミングで何を実施するかを決めておき、確実に OSS を管理する
  - 前頁の承認を受けるタイミング
  - 定期的な確認で OSS のライセンス等に起因する手戻りを削減する

ウォーターフォール型の開発の場合の例



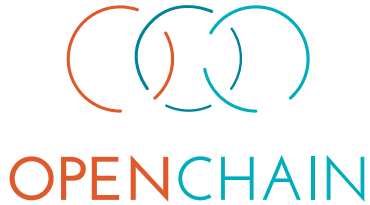
アジャイル型の開発の場合は・・・
- 各スプリントレビューで毎回承認まで行うと回数が多いので、重点的に確認する条件を定めておいたほうがよい
  例: OSS の新規追加、メジャーバージョンアップがあった場合
- または、利用可否判定を自動化する

03

まとめ

OSS の管理とサプライチェーン

- １つの製品には様々な会社/主体が関わり、それぞれがOSSを管理していないと全体として管理できない
  - 開発の委託先
  - OSS の開発元のコミュニティ、ディストリビューター
- OpenChain
  - OSSコンプライアンスの透明性と信頼性を
    サプライチェーン全体で確保することを目指すグローバルコミュニティ
  - 標準・ガイド
    - ISO/IEC 5230: OSS のライセンスコンプライアンスに必要な要件を定義した標準
    - ISO/IEC 18974: OSS のセキュリティ保証に必要な要件を定義した標準
    - SBOM Document Quality Guide (Draft)
  - 日本の OpenChain コミュニティ: OpenChain Japan Work Group



今回の OSS 管理の方針策定と文書化は
5230, 18974 への準拠に必要な活動

まとめ

- 製品には多数の OSS を利用することがあり、この一覧を管理する必要がある
  - ライセンスコンプライアンス
  - セキュリティアシュアランス/脆弱性管理
- OSS の管理のためには、方針を決めてから開発することが大切
  - OSSの情報と成果物の管理方法
    - OSS の情報を記録する方法
    - OSS の成果物を保管する方法
    - OSS の特定方法と範囲
    - 脆弱性管理方法
  - 利用可能ライセンス
  - 体制・プロセス

東芝グループ経営理念

人と、地球の、明日のために。

Committed to People,
Committed to the Future.