修士学位論文

題目

SBOM 利用ツールの評価に向けた Java における SBOM データセットの構築

指導教員
肥後 芳樹 教授

報告者
岸本 理央

令和 7 年 1 月 28 日

大阪大学 大学院情報科学研究科
コンピュータサイエンス専攻 ソフトウェア工学講座

令和 6 年度 修士学位論文

SBOM 利用ツールの評価に向けた Java における SBOM データセットの構築

岸本 理央

内容梗概

ソフトウェアの依存関係の管理において，ソフトウェア部品表（Software Bill of Materials, SBOM）が重要なツールとなっている．SBOM は，ソフトウェアを構成するコンポーネントを一覧して記述したドキュメントであり，コンポーネントの名前やバージョン，ライセンスなどの情報を含む．SBOM を利用することで，ソフトウェアが依存するコンポーネントを素早く特定し，そのコンポーネントが脆弱性を含んでいないかや，コンポーネントのライセンスに違反していないかを確認できる．SBOM を用いたソフトウェアの依存関係管理を支援するツールは多数存在し，それらは SBOM を生成するツールと SBOM を利用するツールに大別できる．SBOM 生成ツールはソフトウェアのソースコードなどを解析し，得られた情報から SBOM を作成するツールである．一方，SBOM 利用ツールは，SBOM を解析し，ソフトウェアの依存関係に関する情報を提供するツールである．SBOM 利用ツールの評価には，正確な情報を含み正しいフォーマットで記述された SBOM が必要であるが，その評価を目的として作成されたデータセットは存在せず，SBOM 利用ツールに関する研究は限られている．この問題を解決するために，本研究では，SBOM 利用ツールの評価に向けた SBOM データセットを構築する．構築したデータセットには，実際の Java プロジェクトから生成された 162 個の SBOM が含まれている．これらの SBOM は，SBOM 利用ツールの評価に必要だと考えられる情報を含むものとなっている．データセットの構築にあたって，GitHub から 3,325 個の Java プロジェクトを収集した．その内 Maven をビルドツールとして利用している 557 個のプロジェクトを対象に，オープンソースの SBOM 生成ツールを利用して SBOM を生成した．生成された SBOM には不足する情報があり，SPDX Lite プロファイルに準拠したものとなっていなかったため，自動修正と手動修正を経て，最終的に 162 個の SBOM を作成した．データセットは Zenodo で公開されており，SBOM 利用ツールの評価に利用可能である（DOI: 10.5281/zenodo.14233415）．作成したデータセットの評価として，データセットに含まれる SBOM の形式が SPDX の規格に準拠していることを確認した．また，作成したデータセットを用いて SBOM 利用ツールの評価実験を行い，既存の SBOM 利用ツールの評価にデータセットが有用であることを確認した．

主な用語

SBOM
SPDX
データセット

目 次

1　はじめに

近年のソフトウェア開発では，多くのライブラリが利用されており，ソフトウェアの依存関係はより複雑化している．ソフトウェアの依存関係の管理が不十分であると，依存するライブラリに脆弱性が発見された場合の対応が遅れることがある [2,8,23]．また，ソフトウェアが依存するライブラリのライセンスを遵守するためには，依存関係を正確に把握する必要がある．ソフトウェアの依存関係を適切に管理するためのツールとして，ソフトウェア部品表（Software Bill of Materials，SBOM）が注目されている [13,22]．SBOM は，ソフトウェアを構成するコンポーネントを一覧して記述したドキュメントであり，コンポーネントの名前やバージョン，ライセンスなどの情報を含む．SBOM を利用することで，ソフトウェアが依存するコンポーネントを素早く特定し，そのコンポーネントに既知の脆弱性が存在しないかを確認したり，ライセンスに違反していないかを確認したりすることができる．

SBOM を用いたソフトウェアの依存関係管理を支援するツールは多数存在し，それらは SBOM を生成するツールと SBOM を利用するツールに大別できる．SBOM 生成ツールはソフトウェアのソースコードなどを解析し，得られた情報に基づいて SBOM を作成するツールである．一方，SBOM 利用ツールは，SBOM を解析し，ソフトウェアの依存関係に関する情報を提供するツールであり，ソフトウェアが依存するライブラリに既知の脆弱性が存在しないかを確認するツールがその一例である．

既存の SBOM 生成ツールの現状や課題は広く研究されている [3,7,19,21,24]．一方，SBOM 利用ツールに関する研究は少ない．この一因として，SBOM 利用ツールの評価に利用できるデータセットが存在しないことが考えられる．SBOM 利用ツールを評価するためには，正確な情報を含み正しいフォーマットで記述された SBOM が必要であるが，その評価を目的として作成されたデータセットは存在しない．SBOM のサンプルファイル [5,16] は，依存関係が少ないソフトウェアに対して作成されているため，SBOM 利用ツールの評価には適していない．また，Chainguard は SBOM のコレクション [4] を提供しているが，これらは自動生成されたりインターネットから取得されたりしたものであり，内容の正確性が保証されていない．

この問題を解決するために，本研究では，SBOM 利用ツールの評価に向けた SBOM データセットを構築する．データセットの構築にあたっては，SBOM の主要なユースケースである脆弱性管理とライセンス管理を想定し，これらの目的に必要な情報を含む SBOM を作成する．SBOM のフォーマットには，SPDX を用い，SPDX Lite プロファイル [18] に準拠した SBOM を作成する．SPDX Lite プロファイルは，産業界での実際の業務フローに基づいて，SBOM に含めるべき情報を定めたものである．このプロファイルに準拠することで，SBOM の主要なユースケースに必要とされる情報を含む SBOM を作成することができる．

GitHub から 3,325 個の Java プロジェクトを収集し，その内 Maven をビルドツールとして利用している 557 個のプロジェクトからオープンソースの SBOM 生成ツールを利用して SBOM を生成した．ツールによって生成された SBOM には不足する情報があり，SPDX Lite プロファイルに準拠していなかったため，自動修正と手動修正を経て，SPDX Lite プロファイルに準拠した 162 個の SBOM を作成した．作成したデータセットの評価として，データセットに含まれる SBOM の形式が SPDX の規格に準拠していることを確認した．また，作成したデータセットを用いて SBOM 利用ツールの評価実験を行い，データセットの有用性を確認した．

以降，2 章では，ソフトウェアの依存関係管理の必要性と SBOM について説明し，SBOM 利用ツールの評価における課題について述べる．3 章では，SBOM 利用ツールの評価に必要なデータセットの要件について述べる．4 章では，SBOM データセットの構築方法について説明する．6 章では，構築したデータセットの評価について述べる．7 章では，SBOM 利用ツールの評価実験について説明する．8 章では，本研究の妥当性への脅威について述べる．最後に，9 章で本研究のまとめと今後の課題について述べる．

2　背景

この章では，本研究の背景として，ソフトウェアの依存関係を管理する必要性と，適切なソフトウェアの管理を行うために利用が推奨されているフトウェア部品表（SBOM）およびSBOM の形式の 1 つである SPDX を説明する．また，SBOM に関するツールの種類とその現状について述べ，SBOM を利用するツールの評価に必要となるデータセットについて述べる．

2.1　ライブラリの利用

ソフトウェアの開発では，ソフトウェアの一部の機能を実現するために既存のライブラリを利用することがある．ライブラリの利用によって，機能の実装に要する時間を削減できるため，開発期間の短縮や開発費用の削減が可能になる．また，広く利用されているライブラリは様々な環境で検証されているため，同様の機能を独自に実装する場合に比べて，信頼性が高く安定しているとされている [10].

ライブラリとして提供されるソフトウェアは，オープンソースソフトウェア（以降単にOSS という）として公開されることがある．ソフトウェア企業の Synopsys が 2022 年に発表した OSS に関する調査報告によると，調査した商用ソフトウェアのコードベースの 97%に OSS が使用されていた [20].

図 1 は，ソフトウェアが依存するライブラリ間の関係を示すグラフの例である．この例では，ソフトウェアがライブラリ A，ライブラリ B，ライブラリ C の 3 つのライブラリに依存している．また，ライブラリ A は，ライブラリ D とライブラリ E に依存しており，ソフトウェアはこれら 2 つのライブラリに間接的に依存している．このように，あるライブラリが他のどのライブラリに依存しているかという依存関係を考慮すると，ソフトウェアが直接依存するライブラリの数が少ない場合でも，全体としては多くのライブラリに依存していることがある．

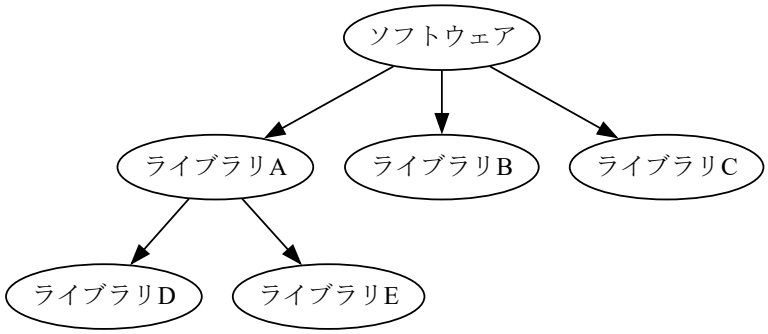

図 1: ソフトウェアの依存関係

2.2　ソフトウェアの依存関係管理の必要性

ソフトウェアの依存関係は，セキュリティとライセンス遵守の 2 つの観点から適切に管理することが重要である．

2.2.1　セキュリティの観点

ライブラリにはセキュリティ上の欠陥である，脆弱性が発見されることがある．ソフトウェアが依存しているライブラリに脆弱性が発見された場合，脆弱性が修正されたバージョンへのバージョンアップなどの対応を速やかに行う必要がある．しかし，深刻な脆弱性であっても，十分な対応が行われないことが問題となっている [2].

2021 年 12 月には Java のログライブラリである Log4j で，リモートコード実行の脆弱性が発見されている．この脆弱性は，脆弱性が存在するバージョンの Log4j が動作しているサーバー等に対して，脆弱性を悪用する細工されたデータを送信することで，攻撃者が任意のコードを実行できる可能性があるというものである．Log4j は汎用のログライブラリであり，Java を用いて開発された様々なソフトウェアで利用されていたことから影響を受けるソフトウェアの数は多く，Maven Central Repository に登録されているパッケージの約 4% を占める，17,000 件以上のパッケージが影響を受けると報告されている [6]．この脆弱性が修正されたバージョンが利用可能になってから 3 年以上が経過している 2025 年 1 月時点においても，直近 1 か月間に行われたダウンロードの 10%以上がこの脆弱性を含むバージョンであることが確認されている [14].

既知の脆弱性はサイバー攻撃の標的となるため，ソフトウェアの依存関係を適切に管理し，依存するライブラリに脆弱性が発見された場合は迅速に対応することが重要である．

2.2.2　ライセンス遵守の観点

ライブラリは，著作権者の著作権で保護されている著作物であり，その使用に当たって従わなければならない条件がソフトウェアライセンスとして指定されている．ライセンスによって課される条件には，著作権表示やライブラリを使用しているソフトウェアのソースコードの公開などが存在する．ソフトウェアが依存する全てのライブラリについて，それぞれのライセンスによって課された条件を満たす必要があるため，ソフトウェアの依存関係を適切に管理し，ソフトウェアが依存するライブラリとそのライセンスを把握することが重要である．

2.3 ソフトウェア部品表（SBOM）

ソフトウェア部品表（Software Bill of Materials，SBOM）は，あるソフトウェアを構成するソフトウェアコンポーネント（ライブラリやファイルなど）について，名前やバージョン，ライセンス，それらの間の依存関係などの情報を一覧して記述したドキュメントである．SBOM に含まれる情報を基に，ソフトウェアが依存するライブラリに脆弱性が発見されていないかや，使用しているライブラリのライセンスに違反していないかを確認できる．近年，ソフトウェアが依存するライブラリに含まれる脆弱性を利用した攻撃が増加していることから，適切なソフトウェアの管理のために SBOM を利用することが推奨されている [13, 22]．日本では経済産業省によって企業による SBOM 導入の手引きが公開されている [25]．アメリカでは，アメリカ合衆国国家電気通信情報管理庁（NTIA）によって SBOM に最低限必要な要素が示されており，SBOM はデータフィールドとして以下の項目を含む必要があるとされている [11]．

- ソフトウェアコンポーネントの提供者
- ソフトウェアコンポーネントの名前
- ソフトウェアコンポーネントのバージョン
- その他の一意な識別子
- ソフトウェアコンポーネント間の依存関係
- SBOM の著者
- SBOM の作成日時

2.4 SBOM のフォーマット

SBOM の主要なフォーマットとして，SPDX [15] と CycloneDX [12] がある．SPDX は Linux Foundation が策定しているフォーマットであり，バージョン 2.2.1 は ISO/IEC 5962:2021 として国際標準化されている．一方，CycloneDX はソフトウェアのセキュリティ分野の研究やガイドラインの作成活動を行っている OWASP が策定しているフォーマットであり，SBOM のみでなく，機械学習モデルに関する BOM である ML-BOM や，IoT デバイスなどのハードウェアに関する BOM である HBOM なども記述できるように設計されている．両フォーマットともにソフトウェアに関連するより多くの情報を記述できるようにするために仕様の改定が継続的に行われている．バージョン 2 系列までの SPDX は SBOM の記述のみをサポートしていたが，2024 年 4 月に，CycloneDX と同様に ML-BOM なども記述できるよう

に仕様が拡張されたバージョン 3.0.0 が公開された．また，CycloneDX も同月にバージョン 1.6 が公開され，仕様の継続的な改善が行われている．

SPDX 形式で記述された，Java プロジェクトの SBOM の一部をコード 1 に示す．この例では，広く利用されている Java のライブラリである log4j-core のバージョン 2.10.0 の情報が記述されている．例えば，ライブラリの名前（*Package Name* フィールド，2 行目），バージョン（*Package Version* フィールド，8 行目），ライセンス（*Concluded License* フィールド，6 行目）がそれぞれのフィールドに記録されている．SPDX は，ソフトウェアに関する様々な種類の情報を記述するために，これら以外にも多くのフィールドを定義しており，CycloneDX でも同様のフィールド群が定義されている．これらのフィールドのほとんどは記述が任意であり，どのフィールドを SBOM に含めるかは想定される SBOM の利用目的によって異なる．

コード 1: SPDX 形式で記述された Java プロジェクトの SBOM の一部

```
1  {
2    "name": "log4j-core",
3    "SPDXID": "SPDXRef-Package-log4net",
4    "downloadLocation": "https://repo1.maven.org/maven2/org/apache/logging/log4j/log4j-cor
         e/2.10.0/log4j-core-2.10.0.jar",
5    "filesAnalyzed": false,
6    "licenseConcluded": "Apache-2.0",
7    "copyrightText": "NOASSERTION",
8    "versionInfo": "2.10.0",
9    "externalRefs": [{
10     "referenceCategory": "PACKAGE-MANAGER",
11     "referenceLocator": "pkg:maven/org.apache.logging.log4j/log4j-core@2.10.0",
12     "referenceType": "purl"
13   }],
14   "supplier": "Organization: The Apache Software Foundation",
15   "homepage": "https://logging.apache.org/ ... "
16 }
```

SPDX では，よく知られたライセンスのリストが SPDX License List [17] として定義されている．SPDX License List では，各ライセンスに SPDX License Identifier と呼ばれる識別子が割り当てられており，SPDX におけるライセンス情報の記述ではこの識別子を使用することが推奨されている．例えば，コード 1 の *Concluded License* フィールド（6 行目）の値である「Apache-2.0」は Apache License 2.0 に対応する SPDX License Identifier である．SPDX License List で定義されていないライセンスについては，そのライセンスを使用する SBOM の中で独自の識別子を用いてライセンスを定義することができる．

2.5 SBOM に関するツール

SBOM を通したソフトウェアの依存関係の管理を支援する様々なツールが存在している．それらのツールは，SBOM を生成するツールと SBOM を消費するツールの 2 種類に大きく分けられる．

SBOM 生成ツールは，ソフトウェアのソースコードなどを解析することで依存するコンポーネントの情報を抽出し，SBOM を作成するツールである．SBOM はソフトウェアの構成要素や，依存関係にある要素から情報を収集して作成するため，手作業で作成するのは容易ではない．これらの負荷を軽減するため，様々な SBOM 生成ツールが存在している [25].

SBOM 利用ツールは，SBOM を解析し，ソフトウェアの依存関係に関する情報を提供するツールであり，SBOM に含まれるデータに基づいて，ソフトウェアが依存するライブラリに脆弱性が発見されていないかを確認するツールや，ライブラリのライセンスに違反していないかを確認するツールなどが含まれる．

2.6 SBOM 利用ツールの評価における課題

SBOM 生成ツールの性能や制約は広く研究されている [3, 7, 19, 21, 24]．一方，SBOM 利用ツールに関する研究は少ない．SBOM 利用ツールの改善を行うためには，ツールを評価するためのベンチマークとなるデータセットがあることが望ましい．例として，コードクローン検出やソフトウェアテストの分野において，ツールやアルゴリズムの評価に用いることができるデータセットが公開されている [1, 26].

公開されている SBOM はあるものの，SBOM 利用ツールを評価することを意図したデータセットは存在していない．Linux Foundation [16] は 13 個のソフトウェアプロジェクトに対して作成した SPDX 形式のサンプルファイルを公開しており，OWASP [5] は 7 個のソフトウェアプロジェクトに対して作成した CycloneDX 形式のサンプルファイルを公開している．これらのサンプルファイルは，依存関係が少ないソフトウェアに対して作成されているため，SBOM 利用ツールの評価には適していない．また，Chainguard [4] は OSS の開発者により公開されている 53 個の SBOM と，人気のある 1,000 個の Docker イメージから複数の SBOM 生成ツールを用いて作成した 3,397 個の SBOM を公開しているが，これらの SBOM は機械的に生成されたり，インターネット上から収集されたりしたものであり，内容の正確性が保証されていない．

3　SBOM 利用ツールの評価のためのデータセットに求められる要件

SBOM の主な利用目的はソフトウェアの脆弱性管理とライセンス管理であり，SBOM 利用ツールの多くがそのための機能を提供している．SBOM 利用ツールの評価を目的としたデータセットを構成する SBOM には，これらの主要な用途に必要な情報が含まれていなければならない．ソフトウェアの脆弱性管理には，ソフトウェアが依存するライブラリの一覧と，それぞれのライブラリを識別する情報が必要である．また，ライセンス管理には，ソフトウェアが依存するライブラリのライセンス情報が必要である．

2.4 節で述べたように，SBOM の主要なフォーマットである SPDX と CycloneDX は，ソフトウェアに関する様々な情報を記述するためのフィールド群を定義している．定義されたフィールドの大部分は任意記述であり，各フォーマットで定義された必須フィールドのみを含む SBOM では，SBOM の主な利用目的に必要とされる情報が満たされない．したがって，データセットを構成する SBOM の作成に当たっては，各フォーマットで定義されてたフィールド群の中から，SBOM の主な利用目的に必要なフィールドを選択する必要がある．

SPDX では，表 1 に示す SPDX Lite プロファイル [18] と呼ばれる必須フィールド群を定義している．SPDX Lite プロファイルは，産業界での実際の業務フローに基づいて SBOM の主な利用目的に必要な情報を含むフィールドを選択したものである．SPDX Lite プロファイルに準拠した SBOM は，SBOM の主な利用目的に必要な情報を含む簡潔な形式であるため，SBOM 利用ツールの評価に適していると考えられる．本論文では，SBOM の形式として SPDX を採用し，SPDX Lite プロファイルに準拠した SBOM を作成することで，SBOM 利用ツールの評価のためのデータセットを構築する．

表 1 に示した SPDX Lite プロファイルに含まれるフィールドの内，L1.1 から L1.7 は SBOM の基本情報を記述するフィールドであり，SBOM の文書自体の識別子や作成者，作成日時などの情報を含む．L2.1 から L2.12 は記述対象のソフトウェアが依存するパッケージ（例：ライブラリ）に関する情報を記述するフィールドであり，ソフトウェアが依存する各パッケージについて，パッケージ名やバージョン，ライセンス情報などが記述される．L3.1 から L3.4 は SPDX License List に記載されていないライセンスの情報を記述するためのフィールドであり，ソフトウェアでそのようなライセンスが使用されている場合は，ライセンスの識別子や条文，名前，ライセンスに関するコメントを記述する．

表 1: SPDX Lite プロファイル

#	フィールド名	フィールドの説明
L1.1	SPDX Version	SPDX フォーマットのバージョン
L1.2	Data License	文書（SBOM）に含まれるデータのライセンス
L1.3	SPDX Identifier	文書の識別子
L1.4	Document Name	文書の名前
L1.5	SPDX Document Namespace	文書の名前空間（文書の識別子）
L1.6	Creator	文書の作成者
L1.7	Created	文書の作成日時
L2.1	Package Name	パッケージ名
L2.2	Package SPDX Identifier	文書内でのパッケージの識別子
L2.3	Package Version	パッケージのバージョン
L2.4	Package File Name	パッケージのファイル名
L2.5	Package Supplier	パッケージの提供者
L2.6	Package Download Location	パッケージのダウンロード場所
L2.7	Files Analyzed	パッケージに含まれるファイルを解析したか
L2.8	Package Home Page	パッケージのホームページ
L2.9	Concluded License	パッケージに適用されるライセンス
L2.10	Declared License	パッケージで宣言されたライセンス
L2.11	Comments on License	ライセンスに関するコメント
L2.12	Copyright Text	著作権者情報
L2.13	Package Comment	パッケージに関するコメント
L2.14	External Reference	外部参照
L3.1	License Identifier	文書内で独自に定義するライセンスの識別子
L3.2	Extracted Text	文書内で独自に定義するライセンスの条文
L3.3	License Name	文書内で独自に定義するライセンスの名前
L3.4	License Comment	ライセンスに関するコメント

4　データセットの構築手法

データセットの構築の流れを図 2 に示す．SBOM の生成対象となるプロジェクトの言語には，様々なソフトウェアプロジェクトで広く利用されており，GitHub 上に多くのプロジェクトが存在する Java を選択した．データセットの構築手順は，Java プロジェクトの収集，既存の SBOM 生成ツールを用いた SBOM の生成，自動修正，手動修正の 4 つのステップに分かれる．最初に，GitHub 上の Java プロジェクトを検索し，見つかったプロジェクトのリポジトリをクローンし，最新のタグをチェックアウトする．次に，オープンソースの SBOM 生成ツールである sbom-tool [9] を用いてプロジェクトのソースコードを解析し，基となる SBOM を生成する．sbom-tool によって作成された SBOM は情報が不足しているため，自動修正のステップで，不足する情報を Java のビルドツールである Maven の設定ファイル（pom.xml）から取得し，SBOM に追加する．最後に，自動修正までの手順で適切な値が設定できなかったフィールドの値を手動で修正し，SPDX Lite プロファイルに準拠した SBOM を作成する．

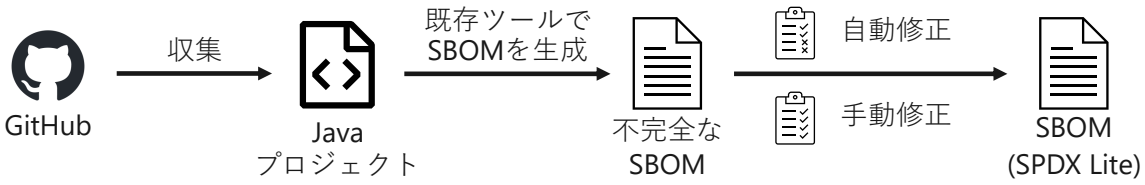

図 2: データセット構築の流れ

4.1　Java プロジェクトの収集

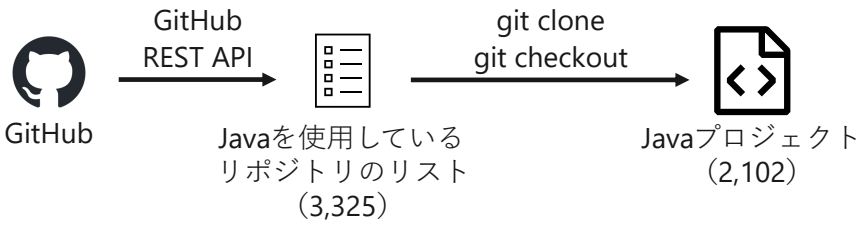

図 3: Java プロジェクトの収集の流れ

SBOM の生成対象となる Java のプロジェクトは GitHub から収集した．最初に，GitHub の REST API を用いて，以下の条件を満たすリポジトリを検索した．

1. 主に使用されている言語が Java である
2. GitHub でのスター数が 50 以上である

検索の結果，これらの条件を満たすリポジトリが 3,325 件見つかった．sbom-tool を用いた SBOM の生成には，生成対象のプロジェクトのソースコードが必要であるため，プロジェクトをクローンした．条件を満たした 3,325 件のリポジトリの内，3,303 件のリポジトリのクローンに成功し，22 件のリポジトリのクローンに失敗した．クローンに失敗した原因は，Windows におけるパスの最大長の制限などであった．

クローンしたリポジトリが初期状態でチェックアウトしているブランチのソースコードは，開発中のコードなど不完全な状態である可能性がある．不完全な状態のソフトウェアを対象として SBOM を生成することを防ぐために，タグ名に snapshot，alpha，beta および rc を含まない最新のタグをチェックアウトした．クローンに成功した 3,303 件のリポジトリの内，2,102 件のリポジトリで条件を満たすタグのチェックアウトに成功した．

4.2 sbom-tool を用いた SBOM の生成

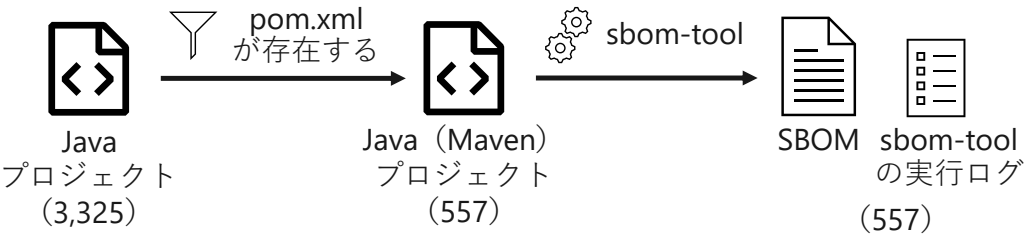

図 4: sbom-tool を用いた SBOM の生成の流れ

リポジトリのクローンに成功し，適切なタグが存在するプロジェクトに対して，sbom-tool [9] を用いて基となる SBOM を生成した．sbom-tool は，Microsoft によってメンテナンスされているオープンソースの SBOM 生成ツールである．このツールは，GitHub 上で 1,700 以上のスターを獲得している人気のツールであり，オープンソースソフトウェアとして開発されていることから，ツールの動作の理解や変更が容易であるため使用した．

最初に，クローンした Java のプロジェクトから Maven をビルドツールとして使用しているプロジェクトを選択した．これは，sbom-tool が，Maven 以外のビルドツールを使用している Java のプロジェクトからの SBOM 生成をサポートしていないためである．Maven は，pom.xml というファイルを用いてビルドに関する設定を行うため，プロジェクトのファイル内に少なくとも 1 つの pom.xml ファイルが存在すれば，Maven をビルドツールとして使用していると判断した．2,102 件のプロジェクトの内，557 件のプロジェクトが Maven をビルドツールとして使用していた．

これらのプロジェクトについて，sbom-tool を用いて SBOM を生成した．sbom-tool は以下のコマンドによって実行した．

sbom−tool generate −b <output−dir> −bc <path−to−src−dir> −pn <name> −pv <version> −ps <supplier>

sbom-tool に与えた引数の説明は以下の通りである．

output-dir	生成される SBOM の出力先ディレクトリのパス
path-to-src-dir	生成対象のプロジェクトのソースディレクトリのパス
name	プロジェクト名（GitHub 上でのリポジトリ名を使用した）
version	プロジェクトのバージョン（最新のタグのコミットハッシュを使用した）
supplier	プロジェクトの提供者（GitHub のリポジトリの所有者名を使用した）

557 件のプロジェクトに対して sbom-tool を実行し，全てのプロジェクトで SBOM の生成に成功した．

なお，sbom-tool では Maven のコンポーネント名に含まれるアルファベットの大文字を小文字に変換する処理が行われていた．Maven のコンポーネント名はケースセンシティブであり，この処理は不適切である．そのため，sbom-tool のバージョン 2.9.9 に，この問題を修正する変更を行い SBOM の生成に使用した．

4.3 自動修正

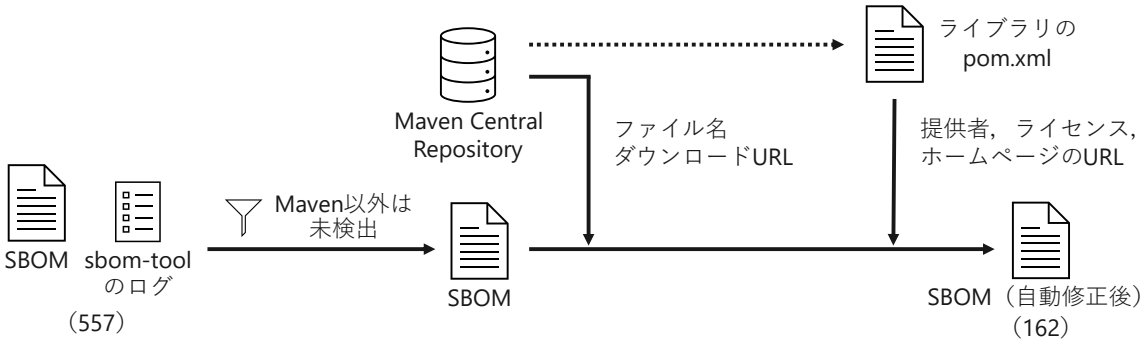

図 5: 自動修正の流れ

sbom-tool では，SPDX Lite プロファイルに含まれているいくつかのフィールドの値が出力されない．表 1 に示した SPDX Lite プロファイルに含まれるフィールドの内，L1.1 から L1.7 の SBOM の基本情報のフィールドは sbom-tool によって埋められるが，L2.1 から L2.12 のパッケージに関する情報は不足している場合がある．また，sbom-tool の出力には

表 2: パッケージに関する必須フィールド

SPDX のフィールド名	sbom-tool	自動修正
Package Name	+	
Package SPDX Identifier	+	
Package Version	+	
Package File Name		+
Package Supplier		o
Package Download Location		+
Files Analyzed	+	
Package Home Page		o
Concluded License		o
Declared License		o
Comments on License		
Copyright Text		
Package Comment		
External Reference	+	

ライセンスの情報が含まれないため，L3.1 から L3.4 の文書内で独自に定義するライセンスの情報は記述されない．

表 2 に，SPDX Lite プロファイルにおけるパッケージ（例：ライブラリ）に関する必須フィールドを示す．1 列目は必須となっている SPDX のフィールド名を示し，2 列目と 3 列目はそのフィールドの値が sbom-tool または自動修正処理によって埋められるかをそれぞれ示している．表中の「+」は全てのプロジェクトにおいてそのフィールドが埋められていることを示し，「o」は情報が取得できた一部のプロジェクトにおいてのみそのフィールドが埋められていることを示す．「sbom-tool」の列が示すように，sbom-tool によって埋められるフィールドは，*Package Name*，*Package SPDX Identifier*，*Package Version*，*Files Analyzed*，*External Reference* の 5 つである．そのため，SPDX Lite に準拠した SBOM を作成するためには，sbom-tool によって埋められなかった残りのフィールドを埋める必要がある．

多くの Java のコンポーネントは Maven Central Repository[1]で公開されている．Maven Central Repository は，REST API[2]を提供しており，この API を通じて残りのフィール

[1]https://central.sonatype.com/
[2]https://central.sonatype.org/search/rest-api-guide/

ドを埋めるために必要な追加の情報を取得することができる．表 3 に，フィールドと API を通して取得可能な情報の対応関係を示す．API を通して取得したコンポーネントの jar ファイル名は Package File Name フィールドの値として，jar ファイルの URL は Package Download Location フィールドの値として使用できる．また，コンポーネントの設定ファイルである pom.xml ファイルへの URL もこの API から得られる．pom.xml ファイルには，そのコンポーネントの提供者やホームページの URL，コンポーネントのライセンスに関する情報が含まれている．Maven Central Repository から得られるこれらの情報を使用することで，*Package File Name*，*Package Supplier*，*Package Download Location*，*Package Home Page*，*Concluded License*，*Declared License* の 6 つのフィールドを埋めることが可能である．

なお，Java プロジェクトの収集時に，検索条件としてプロジェクトで使用されている言語が Java であることを指定しているため，収集されたプロジェクトで主に使用されている言語は Java である．しかし，それらの中には，Java 以外の言語で記述されたコードを含むものが存在する．このことは，そのようなプロジェクトが Maven によって管理されていない依存関係を持つ可能性があることを示している．sbom-tool は，SBOM 生成対象のプロジェクトで使用されているパッケージマネージャーを検出することができ，検出されたパッケージマネージャーの情報は，sbom-tool のログとして出力される．Maven によって管理された依存関係を持つプロジェクトのみを対象とするために，ログの出力内容を確認し，Maven 以外のパッケージマネージャーが検出されたプロジェクトは自動修正の対象から除外した．

自動修正を行うために，Maven Central Repository の API を用いて情報を取得し，対応するフィールドを埋めるスクリプトを作成した．このスクリプトでは Maven Central Repository 以外で公開されているコンポーネントの情報は取得できないため，そのようなコンポーネントに依存するプロジェクトは SBOM 生成の候補から除外した．

sbom-tool で生成された 557 個の SBOM の内，自動修正が成功したものは 162 個であった．

表 3: 自動修正のために Maven Central から取得可能な情報

SPDX のフィールド名	Maven Central から取得可能な情報
Package File Name	jar ファイルの名前
Package Supplier	プロジェクトを運営する組織 および 開発者 (pom.xml)
Package Download Location	jar ファイルの URL
Package Home Page	ホームページの URL (pom.xml)
Concluded License	ライセンス (pom.xml)
Declared License	ライセンス (pom.xml)

4.3.1 Package File Name および Package Download Location

API を通して，コンポーネントの jar ファイル名とその jar ファイルの URL が取得できるため，これらの情報を *Package File Name* フィールドと，*Package Download Location* フィールドの値として，それぞれ使用する．

4.3.2 Package Supplier

開発者は，プロジェクトが所属する組織またはコンポーネントの開発者情報を pom.xml ファイルに任意で記述することができる．pom.xml ファイルが組織情報を含む場合は，*Package Supplier* フィールドにその情報を使用する．組織情報は含まないが開発者情報を含む場合は，*Package Supplier* フィールドにその情報を使用する．複数の開発者の情報を含む場合は，*Package Supplier* フィールドの値として使用する開発者を自動的に決定することはできないため，*Package Supplier* フィールドの値は空欄のままとする．

4.3.3 Package Home Page

開発者は，プロジェクトのホームページの URL を pom.xml ファイルに任意で記述することができる．pom.xml ファイルにホームページの URL が含まれている場合，*Package Home Page* フィールドにその情報を使用する．

4.3.4 Concluded License および Declared License

開発者は，pom.xml ファイルにコンポーネントのライセンスに関する情報を記述することができる．pom.xml ファイルにライセンスの情報が含まれている場合，*Concluded License* フィールドと *Declared License* フィールドにその情報を使用する．

pom.xml ファイルでは，コンポーネントのライセンスが名前とその URL を用いて記述される．pom.xml に記述されたライセンスの名前またはその URL が，SPDX License List に含まれるライセンスの名前またはその URL と完全に一致する場合，そのライセンスの SPDX License Identifier を *Concluded License* フィールドと *Declared License* フィールドの値として使用する．

pom.xml に記述されたライセンスの名前と URL が SPDX License List に含まれるいずれのライセンスとも一致しない場合は，pom.xml に記述されたライセンスの名前をそのまま *Concluded License* フィールドと *Declared License* フィールドの値として使用する．これらについては，手動修正でライセンスの情報を確認して適切な SPDX License Identifier に修正する．

4.4 手動修正

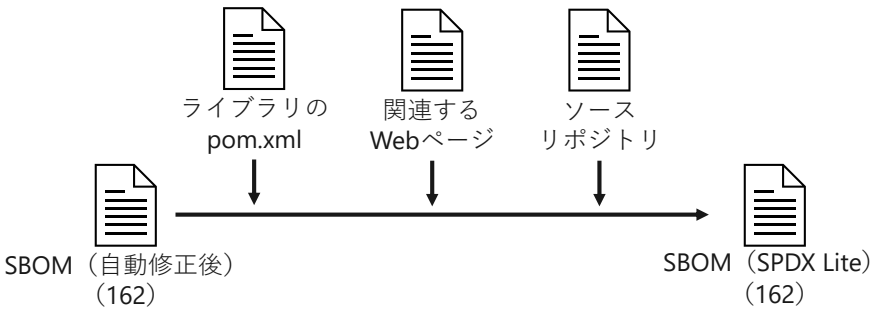

図 6: 手動修正の流れ

自動修正のステップまでで，適切な値を設定できなかったフィールドについては手動修正を行い，SPDX Lite プロファイルに準拠した SBOM を作成する．表 2 に示した必須フィールドの内，*Package Supplier*，*Package Home Page*，*Concluded License*，*Declared License* の 4 つのフィールドは，これらの情報がコンポーネントの pom.xml ファイルに期待される形式で記載されていた場合のみ，自動修正で埋めることができる．これらのフィールドで自動修正に失敗したものは，手動修正によって適切な値を設定する．また，*Copyright Text* フィールドは，pom.xml ファイルに著作権表示を記述する項目が存在せず自動的に適切な値を記述することが難しいため，全てのコンポーネントについて手動修正で適切な値を設定する．なお，本研究で作成する SBOM では，それぞれコンポーネントとそのライセンスに関するコメントを記述するフィールドである *Package Comment* フィールドと *Comments on License* フィールドの値は空文字列とする．自動修正に成功した 162 個の SBOM の 7,349 個のフィールドを手動で修正した．

手動修正の流れを図 6 に示す．手動修正では，フィールドを埋めるために以下の情報源から情報を収集した．

1. コンポーネントの pom.xml ファイル
2. pom.xml ファイルに URL が記載された Web ページ
3. インターネット上を検索して見つかった，コンポーネントに関連する Web ページ

まず，pom.xml ファイルに含まれる情報を確認した．pom.xml ファイルは自動修正でも使用しているが，自動修正のスクリプトでは解析に失敗する pom.xml ファイルが存在したり，XML のコメントとしてライセンスや著作権表示に関する情報が記述されていたりするため，内容を手動で確認した．

加えて，pom.xml ファイルに URL が記述されている Web ページから情報を取得した．pom.xml ファイルには，プロジェクトのホームページの URL，課題管理システムの URL，SourceForge や GitHub などのソースコードリポジトリの URL が記述されていることがある．これらの Web ページから，コンポーネントの提供者やライセンスに関する情報を取得した．URL がリンク切れを起こしている場合，Wayback Machine[3]を用いて Web ページの内容を確認した．

pom.xml ファイルにソースコードリポジトリの URL が記述されておらず，前述の 2 つの情報源から必要な情報を取得できなかった場合は，コンポーネントの名前とバージョンをキーワードとしてインターネット上で検索し，コンポーネントのソースコードリポジトリを発見できた場合はそこから情報を取得した．

いずれの情報源からもフィールドの値を埋めるために必要な情報が得られなかった場合は，SPDX において情報が利用不可であることを示す「NOASSERTION」をフィールドの値とした．

4.4.1 Package Supplier

pom.xml にコンポーネントの提供者が記述されていない場合は，ソースコードリポジトリの所有者情報を *Package Supplier* フィールドの値に使用した．GitHub，GitLab，SourceForge のいずれかのサービスを用いてリポジトリがホスティングされている場合は，リポジトリの所有者名を提供者名とし，所有者のメールアドレスを提供者のメールアドレスとした．これら以外の方法でリポジトリがホスティングされている場合や，ソースコードリポジトリが見つけられなかった場合は，*Package Supplier* フィールドの値は「NOASSERTION」とした．

4.4.2 Package Home Page

pom.xml にホームページの URL が記述されていない場合は，*Package Home Page* フィールドを以下の優先順位で決定される値で埋めた．

1. プロジェクトが GitHub，GitLab，SourceForge でホスティングされている場合は，ホスティングサービス上でプロジェクトの Web サイトとして設定されている URL
2. プロジェクトの README ファイル等にホームページとして記述されている URL
3. インターネット上をプロジェクト名で検索して見つかったホームページの URL
4. ソースコードリポジトリの URL

[3]http://web.archive.org/

これらのいずれの値も取得できなかった場合は，*Package Home Page* フィールドの値は「NOASSERTION」とした．

4.4.3 Concluded License および Declared License

pom.xml にコンポーネントのライセンスが記述されていない場合は，ソースコードリポジトリからライセンスに関する情報を取得した．プロジェクトのライセンス情報は，README ファイルや LICENSE ファイル，NOTICE ファイルに記述されていることがある．これらのファイルがリポジトリに含まれている場合は，そのファイルからライセンスに関する情報を取得し，*Concluded License* フィールドおよび *Declared License* フィールドの値とした．また，プロジェクトのホームページや README ファイル等からリンクされているプロジェクトの Wiki などにライセンス情報が記述されている場合は，その値を使用した．これらの方法でライセンス情報を取得できなかった場合は，*Concluded License* フィールドおよび *Declared License* フィールドの値は「NOASSERTION」とした．

4.4.4 Copyright Text

pom.xml ファイルにはプロジェクトの著作権表示を記述する項目が存在しないが，pom.xml 内に XML コメントとして著作権表示が記述されている場合がある．このような場合は，XML コメント内に記述されている著作権表示を取得し，*Copyright Text* フィールドの値とした．また，プロジェクトの README ファイルや LICENSE ファイル，NOTICE ファイルに著作権表示の記述が存在する場合は，その値を使用した．これらの方法で著作権表示の情報をえられなかった場合は，*Copyright Text* フィールドの値は「NOASSERTION」とした．

5　データセットの構造

作成したデータセットの構造を表 4 に示す．データセットは，データセットに含まれる Java プロジェクトのリストである repositories.json と，プロジェクト毎のデータを保存しているディレクトリ群で構成されている．

表 4: データセットの構造

パス	説明
/repositories.json	データセットに含まれる Java プロジェクトのリスト
/[id]_[name]/	Java プロジェクト毎のディレクトリ
|– sbom.spdx.json	プロジェクトの SBOM のファイル
|– sbom.data-sources.json	SBOM の情報源を記載したファイル

repositories.json において，1 つのプロジェクトに関する情報を記述した部分をコード 2 に示す．リストには，プロジェクトの ID（Id，2 行目），プロジェクト名（FullName，3 行目），プロジェクトの URL（CloneUrl，4 行目），SBOM の生成に使用した Git のコミットハッシュ（CommitHash，5 行目），プロジェクトの依存関係の数（DependenciesCount，6 行目），GitHub でのスター数（StargazersCount，7 行目）が含まれている．プロジェクト毎に，「[プロジェクトの ID]_[プロジェクト名（/は+で置換）]」という名前でディレクトリを作成し，そのディレクトリ内に SBOM ファイル（sbom.spdx.json）と SBOM の情報源を記載したファイル（sbom.data-sources.json）を保存している．

コード 2: repositories.json の記述例

```
1 {
2   "Id": 2700474,
3   "FullName": "alibaba/fastjson",
4   "CloneUrl": "https://github.com/alibaba/fastjson.git",
5   "CommitHash": "26f13f84fdd522de10678e43f55fde918ab7b347",
6   "DependenciesCount": 55,
7   "StargazersCount": 25764
8 },
```

sbom.spdx.json は，そのプロジェクトの SBOM であり，SPDX 2.2 のフォーマットに従って記述され，SPDX Lite プロファイルに準拠している．データセットのユーザーは，repositories.json に記載されている情報を元に SBOM ファイルを選択し，SBOM 利用ツールの評価に使用することができる．

sbom.data-sources.json は，SBOM に情報が記述されているコンポーネントの各フィールドの情報源を記載したファイルである．コード 3 に示すように，コンポーネント毎に，対応する SBOM 内の要素の SPDX ID（SpdxId）と，各フィールドの情報源の種類（DataSourceType），

情報源の名前（Name），および関連する URL（Url）を記述している．このファイルに記載されている情報を参照することで，手動修正された部分について，その修正が正しいかを検証することが可能である．

コード 3: sbom.data-sources.json の記述例

```
{
  "SpdxId": "SPDXRef-Package-EB54CAADD698C71437DCC811430A5A8E9492467F04733D4D52016ED8283B
      C193",
  "Name": {
    "DataSourceType": "Auto",
    "Name": "sbom-tool",
    "Url": "https://github.com/microsoft/sbom-tool/releases/tag/v2.2.9"
  },
  "HomePage": {
    "DataSourceType": "Auto",
    "Name": "Central Repository REST API",
    "Url": "https://search.maven.org/solrsearch/select"
  },
  "ConcludedLicense": {
    "DataSourceType": "Manual",
    "Name": "Corrected manually",
    "Url": "https://github.com/apache/commons-digester/blob/DIGESTER_1_8/LICENSE.txt"
  },
  "CopyrightText": {
    "DataSourceType": "Manual",
    "Name": "Corrected manually",
    "Url": "https://github.com/apache/commons-digester/blob/DIGESTER_1_8/NOTICE.txt"
  },（略）

},
```

6　データセットの評価

作成したデータセットの評価として，データセットを構成する SBOM が SPDX に準拠した正しいフォーマットで記述されていることを確認した．

6.1　評価対象と目的

データセットを構成する SBOM について，SPDX において記述が必須とされている項目の記述漏れが無く，値が記述されている全てのフィールドで，その値が正しい形式で記述されていることを確認する．評価を通して，自動修正のステップまでで自動的に埋められたフィールドでは，sbom-tool や自動修正スクリプトの不具合等に起因する問題が発生していないことを確認する．また，手動修正の対象に含まれる，*Package Supplier*，*Package Home Page*，*Concluded License*，*Declared License* の 4 つのフィールドは，文字列として値を記述するフィールドであるが，記述する文字列のフォーマットが SPDX で定められているため，手動修正の漏れや誤りが発生していないことを確認する．

6.2　検証方法

フォーマットの検証には，SPDX が公式に提供している SPDX Online Tool の Validate 機能[4]を使用した．Validate 機能は，SPDX のファイル形式と SPDX に従って記述された SBOM のファイルを選択して実行することで，選択した SBOM にフォーマット上の問題が無いかを検証することができる．作成した SBOM は，SPDX 2.2 のフォーマットに従って JSON 形式で記述されているため，ファイル形式として「V2 JSON」を選択し，Validate 機能を実行した．検証の実行と結果の取得を自動化するスクリプトを作成し，データセットを構成する 162 個の SBOM 全てに対して Validate 機能を実行した．

6.3　検証結果

作成した 162 個の SBOM 全てについて，エラーや警告は出力されなかった．このことから，全ての SBOM が SPDX に準拠した正しいフォーマットで記述されていることが確認された．

[4]https://tools.spdx.org/app/validate/

7　データセットを用いたSBOM利用ツールの評価実験

作成したデータセットは，SBOM 利用ツールの評価を目的としている．作成したデータセットを用いて既存の SBOM 利用ツールの評価実験を行い，データセットが SBOM 利用ツールの評価に適用可能であることを確認した．評価実験を通して，既存の SBOM 利用ツールの実用上の課題が発見でき，ツールの選択において参考となる情報が得られることを確認した．

7.1　評価対象の SBOM 利用ツール

SPDX の公式ウェブサイト[5]では，SPDX 形式の SBOM に対応する既存の SBOM 利用ツールの一覧が公開されており，提供する機能によって分類されている．この評価実験では，SBOM の内容を手動で確認してソフトウェアの依存関係を管理する作業を行う状況を想定して，Consume/View として分類される，SBOM の閲覧機能を提供するツールを評価の対象とした．23 個のツールが Consume/View として分類されており，この中には実際には SBOM の閲覧機能を持たないツールや，有償のツールが含まれていた．この評価実験では，無償で利用可能で SBOM の閲覧機能を持つことが確認できた，表 5 に示す 5 つのツールを対象に評価を行った．

表 5: 評価対象の SBOM 利用ツール

ツール名	種類	対応フォーマット	使用したバージョン
SUSE[6]	プロプライエタリ	SPDX，CycloneDX	-
bom[7]	オープンソース	SPDX	v0.6.0
Interlynk[8]	プロプライエタリ	SPDX，CycloneDX	-
sbom2doc[9]	オープンソース	SPDX，CycloneDX	v0.5.1
tools-java[10]	オープンソース	SPDX	v2.0.0-RC10.5.1

7.1.1　SUSE

SUSE は，SPDX 形式または CycloneDX 形式で記述された SBOM の閲覧機能を提供する Web アプリケーションである．閲覧したい SBOM ファイルを選択することで，SBOM の内

[5]https://spdx.org/tools/
[6]https://apps.rancher.io/sbom-viewer
[7]https://github.com/kubernetes-sigs/bom
[8]https://app.interlynk.io/
[9]https://pypi.org/project/sbom2doc/
[10]https://github.com/spdx/tools-java

容を閲覧することができる．

7.1.2 bom

bom は，SPDX 形式で記述された SBOM の閲覧機能を提供する CLI ツールである．入力として与えられた SBOM に記述されたコンポーネントの一覧を表示する document outline コマンドと，指定した条件に一致するコンポーネントを検索する document query コマンドを提供している．これらのコマンドは，SBOM ファイルのパスである<path>を引数として，以下のコマンドで実行する．

bom document outline <path>

bom document query <path> ”depth:1” −−fields name,version,license,supplier,originator,url

7.1.3 Interlynk

Interlynk はソフトウェアの依存関係管理を行える Web アプリケーションである．SPDX 形式または CycloneDX 形式で記述された SBOM をインポートして，その情報に基づいてソフトウェアが依存するライブラリに脆弱性が含まれていないかを確認したり，使用されているライセンスの情報を確認したりすることができる．

7.1.4 sbom2doc

sbom2doc は，SPDX 形式または CycloneDX 形式で記述された SBOM に記述されたコンポーネントの情報を，様々な形式の文書に変換して出力する CLI ツールである．出力形式は，コンソール出力，Excel ファイル，HTML ファイル，JSON ファイル，Markdown ファイル，PDF ファイルのいずれかを選択することができる．本評価実験では，出力形式として HTML ファイルを選択した．sbom2doc は，SBOM ファイルのパスである<path>と出力形式である<format>，出力先のファイルパスである<output>を引数として，以下のコマンドで実行する．

sbom2doc −−input−file <path> −−format <format> −−output−file <output>

7.1.5 tools-java

tools-java は，SPDX の規格を策定している SPDX Workgroup が公式に提供するツールである．SPDX 形式の SBOM の作成や閲覧を行うための様々なコマンドを提供している．本評価実験では，SBOM の閲覧機能について評価を行った．tools-java は，SBOM ファイルのパスである<path>を引数として，以下のコマンドで実行する．

java −jar tools−java−2.0.0−RC1−jar−with−dependencies.jar SPDXViewer <path>

7.2 評価に使用した SBOM

本評価実験の目的は，データセットが SBOM 利用ツールの課題の発見に役立つかを確認することであり，既存ツールの問題点を網羅的に明らかにすることは目的としていない．そのため，データセットに含まれる 162 個の SBOM から 7 個を選択し，評価に使用した．スター数が多く広く利用されていると考えられるプロジェクトに加えて，複雑な依存関係を持つプロジェクトや，SPDX License List に含まれないライセンスが使用されているプロジェクトなど，特に問題を引き起こしやすいと考えられるプロジェクトの SBOM を選択した．

選択した SBOM のプロジェクト情報，GitHub 上でのスター数，依存の数，SPDX License List に含まれていないライセンス（LicenseRef を用いたライセンス表記）の有無を表 6 に示す．graphware/neo4j-nlp は最も依存するコンポーネントの数が多いため，複雑な依存関係を持つ SBOM として評価に使用した．alibaba/fastjson と square/javapoet はデータセットに含まれるプロジェクトのスター数上位 2 プロジェクトであり，人気があり広く利用されているプロジェクトであることから選択した．konik-io/konik と mojohaus/rpm-maven-plugin はスター数と依存の数がともに少ないプロジェクトであるが，ライセンス情報を記述する *Declared License* フィールドと *Concluded License* フィールドが異なる値を持っており，そのような SBOM の表示について評価を行うために選択した．また，update4j/update4j と square/haha は，依存の数が少なく単純な依存関係を持つ SBOM として評価に使用した．

表 6: SBOM 利用ツールの評価に使用した SBOM

No.	ID	プロジェクト名	スター数	依存の数	LicenseRef
1	56810670	graphware/neo4j-nlp	336	250	あり
2	2700474	alibaba/fastjson	25,764	55	あり
3	7961991	square/javapoet	10,838	19	なし
4	24074294	konik-io/konik	51	54	あり
5	35699733	mojohaus/rpm-maven-plugin	57	49	あり
6	124780056	update4j/update4j	794	1	なし
7	34395232	square/haha	1,448	3	あり

7.3 評価方法

各 SBOM を入力として SBOM 利用ツールの実行に成功するかを確認し，与えられた SBOM を正しく解析できるかを評価した．実行に成功した場合は，ツールの出力に含まれる情報を目視で確認し，SPDX Lite プロファイルに含まれるフィールドの情報が出力に含まれてい

るかを評価した.

7.4 評価結果

表6に示したSBOMを入力として，各ツールの実行に成功したかを表7に示す．表中の「o」は実行に成功したことを示し,「x」は実行に失敗したことを示す．SUSE，bom，sbom2doc，tools-javaの4つのツールは全てのSBOMで実行に成功した．一方，Interlynkの実行に成功したのはNo.6のupdate4j/update4jを入力として与えたときのみであり，他の6つのSBOMでは実行に失敗した.

表 7: SBOM 利用ツールの実行結果

ツール	SBOM の No.						
	1	2	3	4	5	6	7
SUSE	o	o	o	o	o	o	o
bom	o	o	o	o	o	o	o
Interlynk	x	x	x	x	x	o	x
sbom2doc	o	o	o	o	o	o	o
tools-java	o	o	o	o	o	o	o

また，依存するコンポーネントの情報として，SBOMのどのフィールドの情報がツールの出力に含まれるかを表8に示す．表中の「o」はそのフィールドが出力に含まれていることを,「p」は部分的に出力に含まれることを,「x」は不正確な値が出力に含まれることを示す．パッケージ名とパッケージのバージョンは全てのツールの出力に含まれていたが，その他のフィールドはツールによって出力に含まれない場合や情報が不正確な場合があった.

7.4.1 SUSE

図7にSUSEの実行結果の例を示す．SUSEは全てのSBOMで実行に成功し，出力には*Package Name*，*Package SPDX Identifier*，*Package Version*，*Declared License*の4つのフィールドの情報が含まれた．なお，SPDX License Listに含まれていないライセンスの情報については，ドキュメント内での識別子（LicenseRef-で始まる文字列）のみが表示され，そのライセンスの詳細を確認する方法は提供されていなかった.

表 8: ツールの出力に含まれる情報

SPDX のフィールド名	ツール				
	SUSE	bom	Interlynk	sbom2doc	tools-java
Package Name	o	o	o	o	o
Package SPDX Identifier	o				o
Package Version	o	o	o	o	o
Package File Name					o
Package Supplier		x	p	o	o
Package Download Location		o		o	o
Files Analyzed					o
Package Home Page					o
Concluded License			(o)	p	o
Declared License	p	p			o
Copyright Text				o	o
External Reference			o	o	o

7.4.2 bom

bom の実行結果の例をコード 4 に示す．bom は全ての SBOM で実行に成功し，出力には *Package Name*，*Package Version*，*Package Download Location*，*Declared License* の 4 つのフィールドの情報が含まれた．*Package Supplier* フィールドの値は常に「_」と表示されており，正しい値は表示されなかった．また，SPDX License List に含まれていないライセンスの情報については，ドキュメント内での識別子（LicenseRef-で始まる文字列）のみが表示され，そのライセンスの詳細を確認する方法は提供されていなかった．

7.4.3 Interlynk

Interlynk の実行結果の例を図 8 に示す．Interlynk は No.6 の update4j/update4j を入力として与えたときのみ実行に成功し，他の 6 つの SBOM では実行に失敗した．実行に失敗した場合は，コード 5 に示すエラーメッセージが出力された．

実行成功時の出力には *Package Name*，*Package Version*，*Package Supplier*，*Concluded License*，*External Reference* の 5 つのフィールドの情報が含まれた．*Package Supplier* フィールドの値は，パッケージの提供者の種類が組織の場合のみ表示され，個人の場合は表示されなかった．なお，実行に成功した No.6 の SBOM には，SPDX License List に含まれていな

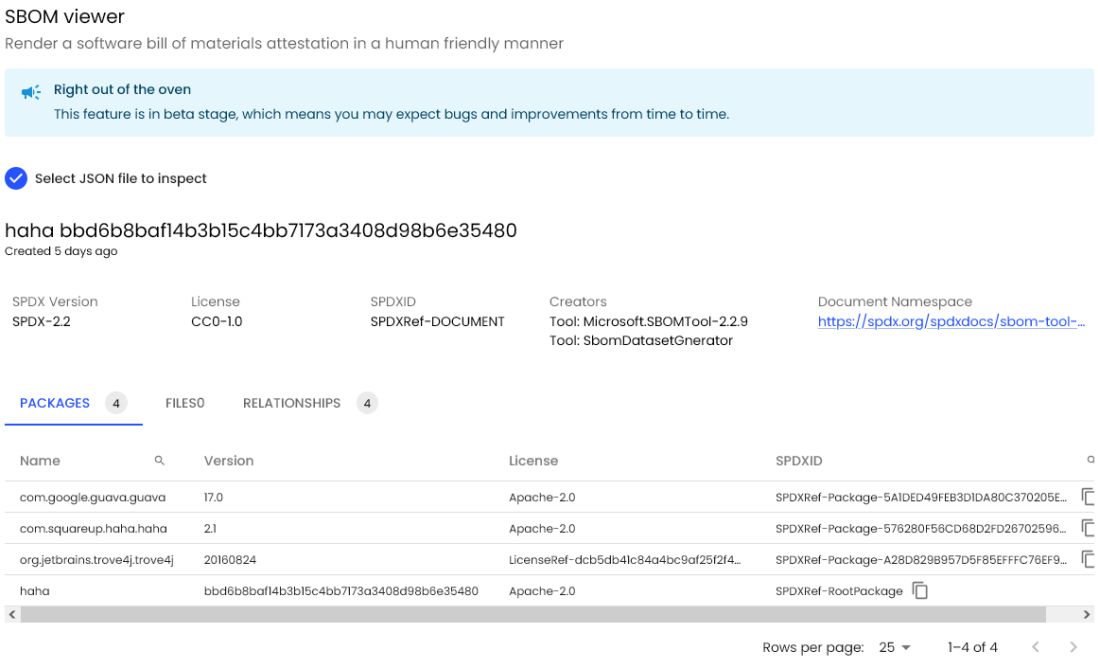

図 7: SUSE の実行結果（No.7 square/haha）

コード 4: bom の実行結果（No.7 square/haha）

```
1 com.google.guava.guava 17.0 Apache-2.0 _ _ https://repo1.maven.org/maven2/com/google/guava/gua
      va/17.0/guava-17.0.jar
2 org.jetbrains.trove4j.trove4j 20160824 LicenseRef-dcb5db41c84a4bc9af25f2f48ab417d9 _ _ http
      s://repo1.maven.org/maven2/org/jetbrains/trove4j/trove4j/20160824/trove4j-20160824.jar
3 com.squareup.haha.haha 2.1 Apache-2.0 _ _ https://repo1.maven.org/maven2/com/squareup/haha/hah
      a/2.1/haha-2.1.jar
```

いライセンスの情報が含まれないため，そのようなライセンス情報の表示については確認できなかった．

7.4.4 sbom2doc

sbom2doc の実行結果の例を図 9 に示す．sbom2doc は全ての SBOM で実行に成功し，出力には *Package Name*，*Package Version*，*Package Supplier*，*Package Download Location*，*Concluded License*，*Copyright Text*，*External Reference* の 7 つのフィールドの情報が含まれた．なお，SPDX License List が含まれていないライセンスの情報については，ドキュメント内での識別子（LicenseRef-で始まる文字列）のみが表示され，そのライセンスの詳細を確認する方法は提供されていなかった．

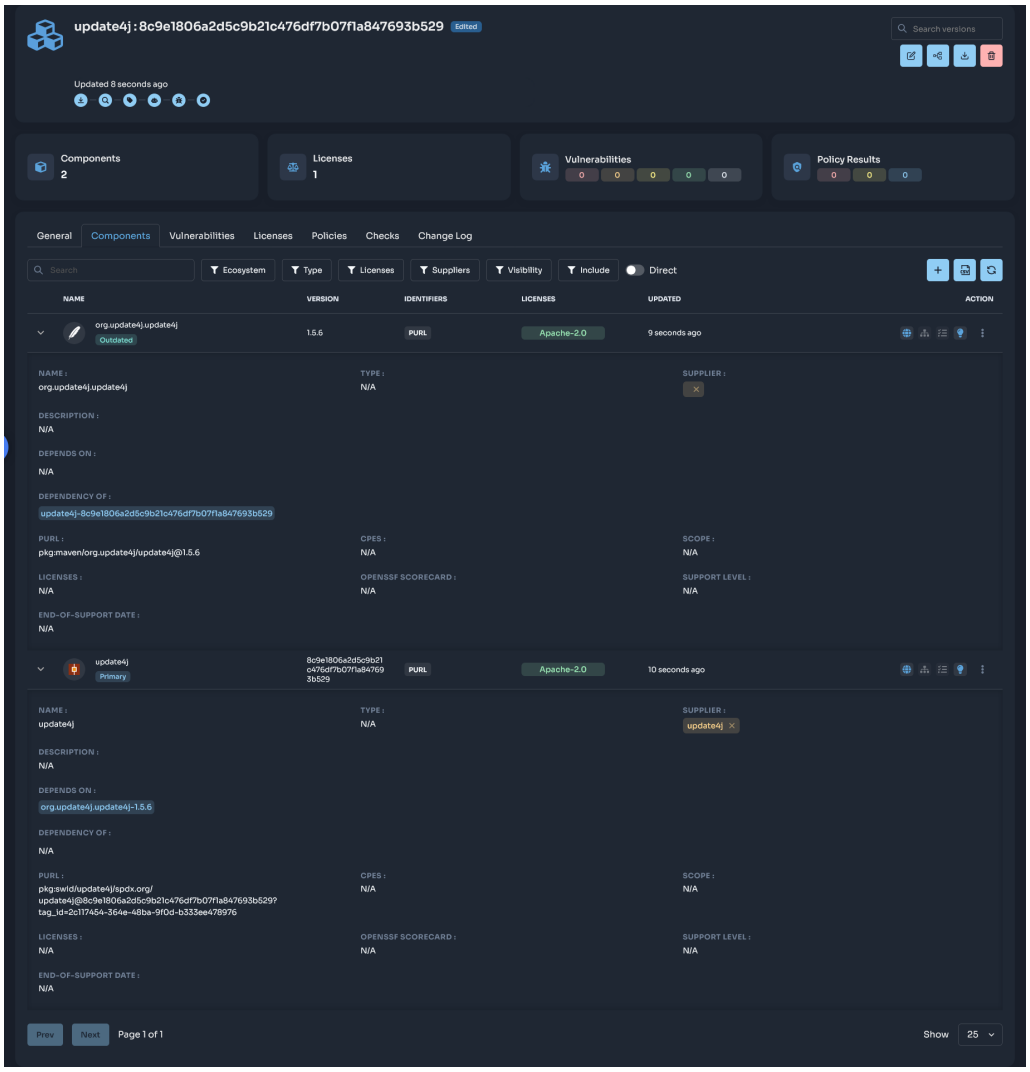

図 8: Interlynk の実行結果（No.6 update4j/update4j）

7.4.5 tools-java

tools-java の実行結果の例をコード 6 に示す．tools-java は全ての SBOM で実行に成功し，出力には SBOM に記述された全てのフィールドの情報が含まれた．SPDX License List に含まれていないライセンスの情報についても，その詳細が表示されていた．

7.5 既存の SBOM 利用ツールの課題

ツールによって，SBOM に含まれるどの情報を表示するかは異なった．依存するライブラリの名前（*Package Name* フィールド）とバージョン（*Package Version* フィールド）は全てのツールで表示されたが，提供者（*Package Supplier* フィールド）やダウンロード URL

コード 5: Interlynk のエラーメッセージ

```
1  sbom / Error processing SBOM sbom.spdx.json for project default: All objects being inserted mu
       st have the same keys}
```

（*Package Download Location* フィールド），著作権表示（*Copyright Text* フィールド），外部参照（*External Reference*）の情報は一部のツールでのみ表示された．SPDX Lite プロファイルに含まれ，SBOM の主な利用目的に必要であると考えられる情報であっても，それらを表示しないツールが多かった．

ライセンスに関しては，tools-java を除いて，*Concluded License* フィールドと *Declared License* フィールドのいずれか一方の情報のみが表示された．一方しか表示しない場合にどちらのフィールドの情報を表示するかはツールによって異なっており，いずれのツールもこの動作を明確に文書化していなかった．また，SPDX License List に含まれていないライセンスの情報については，その詳細が表示されないツールが多く，サポートが不十分であることが分かった．

また，プロプライエタリなツールである Interlynk は，7 個中 6 個の SBOM で実行に失敗した．実行失敗時のエラーメッセージの内容からエラーの原因を特定することは難しいが，Interlynk の実行に失敗した SBOM を他のツールに入力として与えた場合は実行に成功しているため，Interlynk の SBOM の解析処理に問題が存在する可能性がある．

入力された SBOM に含まれる全ての情報が確認できるツールは，tools-java のみであった．しかし，tools-java は，SBOM に含まれる情報を標準出力にテキストとして出力するため，複雑な依存関係を持つソフトウェアの場合は情報の閲覧と検索が困難である．sbom2doc のように情報の閲覧や検索が容易な出力形式を提供し，かつ tools-java のように情報の不足が無いツールが必要であると考えられる．

7.6 考察

作成したデータセットは SBOM 利用ツールの評価と改善に役立つものであると考えられる．データセットに含まれる 7 個の SBOM を入力として既存の SBOM 利用ツールを実行した結果から，既存ツールの課題を発見することができた．データセットには，その他にも様々なプロジェクトの SBOM が含まれており，それらを利用して SBOM 利用ツールの評価を行うことで，より広範に課題を明らかにすることができると考えられる．

SBOM Summary

Item	Details
SBOM File	E:¥lab¥spdx¥mthesis-sbom-dataset¥data¥publish¥34395232_square+haha¥sbom.spdx.json
SBOM Type	spdx
Version	SPDX-2.2
Name	haha bbd6b8baf14b3b15c4bb7173a3408d98b6e35480
Creator	Tool: Microsoft.SBOMTool-2.2.9
Creator	Tool: SbomDatasetGnerator
Created	2025-01-21T06:55:18Z
Files	0
Packages	4
Relationships	4
Services	0
Vulnerabilities	0

Package Summary

Name	Version	Type	Supplier	License
com.google.guava.guava	17.0	LIBRARY	Kevin Bourrillion (kevinb@google.com)	Apache-2.0
com.squareup.haha.haha	2.1	LIBRARY	Square, Inc.	Apache-2.0
org.jetbrains.trove4j.trove4j	20160824	LIBRARY	JetBrains Team (trove4j@jetbrains.com)	NOASSERTION
haha	bbd6b8baf14b3b15c4bb7173a3408d98b6e35480	LIBRARY	square	Apache-2.0

Name	Version	Ecosystem	Download	Copyright
com.google.guava.guava	17.0	maven	https://repo1.maven.org/maven2/com/google/guava/guava/17.0/guava-17.0.jar	NOASSERTION
com.squareup.haha.haha	2.1	maven	https://repo1.maven.org/maven2/com/squareup/haha/haha/2.1/haha-2.1.jar	NOASSERTION
org.jetbrains.trove4j.trove4j	20160824	maven	https://repo1.maven.org/maven2/org/jetbrains/trove4j/trove4j/20160824/trove4j-20160824.jar	NOASSERTION
haha	bbd6b8baf14b3b15c4bb7173a3408d98b6e35480	swid	NOT KNOWN	NOASSERTION

Name	PURL	CPE
com.google.guava.guava	pkg:maven/com.google.guava/guava@17.0	
com.squareup.haha.haha	pkg:maven/com.squareup.haha/haha@2.1	
org.jetbrains.trove4j.trove4j	pkg:maven/org.jetbrains.trove4j/trove4j@20160824	
haha	pkg:swid/square/spdx.org/haha@bbd6b8baf14b3b15c4bb7173a3408d98b6e35480?tag_id=69551956-f4de-46cf-8a8b-34bcc13a3a79	

Component Type Summary

Type	Count
LIBRARY	4

License Summary

License	Count
Apache-2.0	3
NOASSERTION	1

Supplier Summary

Supplier	Count
JetBrains Team (trove4j@jetbrains.com)	1
Kevin Bourrillion (kevinb@google.com)	1
Square, Inc.	1
square	1

NTIA Summary

Element	Status
All file information provided?	True
All package information provided?	True
Creator identified?	True
Creation time identified?	True
Dependency relationships provided?	True

NTIA conformant True

図 9: sbom2doc の実行結果（No.7 square/haha）

8　データセットの妥当性への脅威

8.1　自動修正における誤りの可能性

データセットの作成において，既存の SBOM 生成ツールによって作成した SBOM を，Maven Central Repository から取得した情報を用いて自動修正した．自動修正では，取得した情報が正確であることを前提としている．取得した情報の多くはコンポーネントの開発者が記述した pom.xml ファイルの内容に基づいているため，pom.xml ファイルに誤った情報が記述されている場合，自動修正によって誤った情報が SBOM に取り込まれる可能性がある．例えば，プロジェクトが他のプロジェクトのフォークであり，pom.xml ファイルが更新されていない場合，コンポーネントの提供者やプロジェクトのホームページの情報はオリジナルのプロジェクトの情報となり，誤った情報を含む SBOM が自動修正によって作成さ

コード 6: tools-java の実行結果（No.7 square/haha）

```
Version: SPDX-2.2
Data License: CC0-1.0
Document Namespace: https://spdx.org/spdxdocs/sbom-tool-2.2.9-bf9b6ad7-b93c-43f9-a45d-b2523d6a3d63/haha/bbd6b8baf14b3b15c4bb7173a3408d98b6e35480/AlJy7mNwV0KxE1LMYGuzzA
Document Name: haha bbd6b8baf14b3b15c4bb7173a3408d98b6e35480
ID: SPDXRef-DOCUMENT

Created by:
        Tool: Microsoft.SBOMTool-2.2.9
        Tool: SbomDatasetGnerator
        2025-01-21T06:55:18Z
Relationships:
        Relationship: SPDXRef-DOCUMENT DESCRIBES SPDXRef-RootPackage


Package Name: haha
ID: SPDXRef-RootPackage
Version: bbd6b8baf14b3b15c4bb7173a3408d98b6e35480
File name: ./
Supplier: Organization: square
SPDX Document for :git+https://github.com/square/haha.git
Home Page: https://github.com/square/haha
License concluded: Apache-2.0
License declared: Apache-2.0
Declared Copyright: NOASSERTION
External Reference: PACKAGE-MANAGER purl pkg:swid/square/spdx.org/haha@bbd6b8baf14b3b15c4bb7173a3408d98b6e35480?tag_id=69551956-f4de-46cf-8a8b-34bcc13a3a79
Files Analyzed: false
Relationships:
        Relationship: SPDXRef-RootPackage DEPENDS_ON SPDXRef-Package-576280F56CD68D2FD267025960ECB37F7AFD0B1C60D68BAD28A163ED96592C58
        Relationship: SPDXRef-RootPackage DEPENDS_ON SPDXRef-Package-5A1DED49FEB3D1DA80C370205EDEA29074AE1C3605CB3622107F38027B6919BC
        Relationship: SPDXRef-RootPackage DEPENDS_ON SPDXRef-Package-A28D829B957D5F85EFFFC76EF95481173BA2CFD96E4C76970DE7F073F37BA291


Package Name: com.google.guava.guava
ID: SPDXRef-Package-5A1DED49FEB3D1DA80C370205EDEA29074AE1C3605CB3622107F38027B6919BC
Version: 17.0
File name: guava-17.0.jar
Supplier: Person: Kevin Bourrillion (kevinb@google.com)
SPDX Document for :https://repo1.maven.org/maven2/com/google/guava/guava/17.0/guava-17.0.jar
Home Page: http://code.google.com/p/guava-libraries
License concluded: Apache-2.0
License declared: Apache-2.0
Declared Copyright: NOASSERTION
External Reference: PACKAGE-MANAGER purl pkg:maven/com.google.guava/guava@17.0
Files Analyzed: false


Package Name: com.squareup.haha.haha
ID: SPDXRef-Package-576280F56CD68D2FD267025960ECB37F7AFD0B1C60D68BAD28A163ED96592C58
Version: 2.1
File name: haha-2.1.jar
Supplier: Organization: Square, Inc.
SPDX Document for :https://repo1.maven.org/maven2/com/squareup/haha/haha/2.1/haha-2.1.jar
Home Page: https://github.com/square/haha
License concluded: Apache-2.0
License declared: Apache-2.0
Declared Copyright: NOASSERTION
External Reference: PACKAGE-MANAGER purl pkg:maven/com.squareup.haha/haha@2.1
Files Analyzed: false


Package Name: org.jetbrains.trove4j.trove4j
ID: SPDXRef-Package-A28D829B957D5F85EFFFC76EF95481173BA2CFD96E4C76970DE7F073F37BA291
Version: 20160824
File name: trove4j-20160824.jar
Supplier: Person: JetBrains Team (trove4j@jetbrains.com)
SPDX Document for :https://repo1.maven.org/maven2/org/jetbrains/trove4j/trove4j/20160824/trove4j-20160824.jar
Home Page: https://github.com/JetBrains/intellij-deps-trove4j
License concluded: NOASSERTION
License declared: LicenseRef-dcb5db41c84a4bc9af25f2f48ab417d9
Declared Copyright: NOASSERTION
External Reference: PACKAGE-MANAGER purl pkg:maven/org.jetbrains.trove4j/trove4j@20160824
Files Analyzed: false


Non-Standard Licenses:
        License ID: LicenseRef-dcb5db41c84a4bc9af25f2f48ab417d9
        Text: https://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html
        License Name: https://www.gnu.org/licenses/old-licenses/lgpl-2.1.en.html
```

れてしまう．

本研究のデータセットの作成手法では，*Copyright Text* フィールドが，全てのコンポーネントについて手動修正の対象となっているため，全てのコンポーネントについて pom.xml ファイルの内容を手動で確認している．手動修正の過程で前述の例のような誤りを含む例は確認されなかったため，自動修正によって誤った情報が SBOM に取り込まれている可能性は低いと考えられるが，手動修正で見落とされている誤りが存在する可能性は否定できない．

8.2 手動修正における誤りの可能性

自動修正後の SBOM において不足している情報は，手動で修正を行った．手動修正作業は，著者 1 人によって行われたため，手動修正された部分には誤りが含まれる可能性がある．本研究では，データセットの信頼性を高めるために，SBOM に含まれるデータの取得元を記録し，各プロジェクトの SBOM のファイルとともに提供している．SBOM の利用者は手動修正におけるデータの取得元を参照し，SBOM の正確性を検証することができる．

8.3 データセットの制約と有用性

本研究で作成したデータセットは，162 個の SBOM から構成されている．データセットに含まれる SBOM の数が限定されているため，SBOM 利用ツールの包括的な評価には不十分である可能性がある．データセットに含まれる SBOM は，実際のソフトウェアプロジェクトから作成されたものである．作成対象のプロジェクトには，GitHub 上で多くのスターを獲得しているプロジェクトや，複雑な依存関係を持つプロジェクトが含まれているため，特定のシナリオでの SBOM 利用ツールの評価においては有用であると考えられる．7 節で述べた評価実験において，SBOM の閲覧機能を有するツールの評価を行った結果，評価対象となったツールの課題が見つかったことから，データセットはそのようなツールの評価に適用可能であることが確認されている．

9　おわりに

本研究では，SBOM 利用ツールの評価に向けて SBOM のデータセットを構築した．データセットは，GitHub から収集した実際の Java プロジェクトから生成された 162 個の SBOM から構成されている．これらの SBOM は，オープンソースの SBOM 生成ツールである sbom-tool を用いて生成され，自動修正と手動修正を経て SPDX Lite プロファイルに準拠したものとなっており，SBOM の主な利用目的に必要な情報を含む．

また，作成したデータセットを用いて，既存の SBOM 利用ツールの評価実験を行った．評価実験によって既存の SBOM 利用ツールの課題が明らかになり，作成したデータセットが SBOM 利用ツールの評価に有用であることが確認された．

今後の課題としては，データセットの拡張が挙げられる．本研究では，Java プロジェクトを対象として SBOM を作成したが，他のプログラミング言語で書かれたプロジェクトの SBOM も含めることで，SBOM 利用ツールの包括的な評価を行うことができると考えられる．

謝辞

大阪大学大学院情報科学研究科コンピュータサイエンス専攻 肥後 芳樹 教授には，ご多忙の中研究活動に対して多くの貴重なご助言やご指導を賜りました．心より深く感謝申し上げます．大阪大学大学院情報科学研究科コンピュータサイエンス専攻 Raula Gaikovina Kula 教授，ならびに松下 誠 准教授には，研究室の発表機会において，ご意見，ご助言を賜りました．心より深く感謝申し上げます．

南山大学大学院理工学研究科ソフトウェア工学専攻 井上 克郎 教授，福知山公立大学情報学部 眞鍋雄貴 講師，ならびにノートルダム清心女子大学情報デザイン学部情報デザイン学科 神田 哲也 准教授には研究活動の直接のご指導，論文の執筆に至るまで，あらゆる場面で多くのご指導を賜りました．心より深く感謝申し上げます．

また，株式会社東芝 デジタルイノベーションテクノロジーセンター 仇 実 氏には，研究の着想から論文執筆に至るまで，実務者の観点から多くのご助言を賜りました．心より深く感謝申し上げます．

最後に，その他様々なご指導，ご助言等を賜りました，大阪大学大学院情報科学研究科コンピュータサイエンス専攻肥後研究室の皆様に，心より深く感謝申し上げます．

参考文献

[1] Farouq Al-Omari, Chanchal K. Roy, and Tonghao Chen. Semanticclonebench: A semantic code clone benchmark using crowd-source knowledge. In *2020 IEEE 14th International Workshop on Software Clones (IWSC)*, pp. 57–63, 2020.

[2] Mahmoud Alfadel, Diego Elias Costa, and Emad Shihab. Empirical Analysis of Security Vulnerabilities in Python Packages. In *Proceedings of the 2021 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 446–457, 2021.

[3] Musard Balliu, Benoit Baudry, Sofia Bobadilla, Mathias Ekstedt, Martin Monperrus, Javier Ron, Aman Sharma, Gabriel Skoglund, César Soto-Valero, and Martin Wittlinger. Challenges of Producing Software Bill of Materials for Java. *IEEE Security & Privacy*, pp. 2–13, 2023.

[4] Chainguard. bom-shelter. `https://github.com/chainguard-dev/bom-shelter`, 2022.

[5] CycloneDX. bom-examples. `https://github.com/CycloneDX/bom-examples`.

[6] Google. Understanding the Impact of Apache Log4j Vulnerability. `https://security.googleblog.com/2021/12/understanding-impact-of-apache-log4j.html`, 2021. 2025 年 1 月 28 日に閲覧.

[7] Andreas Halbritter and Dominik Merli. Accuracy evaluation of sbom tools for web applications and system-level software. In *Proceedings of the 19th International Conference on Availability, Reliability and Security*, ARES '24, New York, NY, USA, 2024. Association for Computing Machinery.

[8] Raula Gaikovina Kula, Daniel M. German, Ali Ouni, Takashi Ishio, and Katsuro Inoue. Do developers update their library dependencies? *Empirical Software Engineering*, Vol. 23, No. 1, pp. 384–417, February 2018.

[9] Microsoft. sbom-tool. 2025 年 1 月 28 日に閲覧.

[10] P. Mohagheghi, R. Conradi, O.M. Killi, and H. Schwarz. An empirical study of software reuse vs. defect-density and stability. In *Proceedings. 26th International Conference on Software Engineering*, pp. 282–291, 2004.

[11] NTIA. The Minimum Elements For a Software Bill of Materials (SBOM). https://www.ntia.gov/report/2021/minimum-elements-software-bill-materials-sbom, 2021.

[12] OWASP Foundation. OWASP CycloneDX Software Bill of Materials (SBOM) Standard. https://cyclonedx.org/.

[13] The European Parliament. Cyber resilience act. https://www.europarl.europa.eu/doceo/document/TA-9-2024-0130_EN.html, 2024.

[14] Sonatype. Log4j Updates and Vulnerability Resources. https://www.sonatype.com/resources/log4j-vulnerability-resource-center. 2025年1月28日に閲覧.

[15] SPDX Workgroup. About - Software Package Data Exchange (SPDX). https://spdx.dev/about.

[16] SPDX Workgroup. spdx-examples. https://github.com/spdx/spdx-examples.

[17] SPDX Workgroup. Spdx license list. https://spdx.github.io/spdx-spec/v2.2.2/SPDX-license-list/.

[18] SPDX Workgroup. Spdx lite. https://spdx.github.io/spdx-spec/v2.3/SPDX-Lite/.

[19] Trevor Stalnaker, Nathan Wintersgill, Oscar Chaparro, Massimiliano Di Penta, Daniel M German, and Denys Poshyvanyk. Boms away! inside the minds of stakeholders: A comprehensive study of bills of materials for software systems. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE)*, pp. 513–525, 2024.

[20] Synopsys. 2022 Open Source Security and Risk Analysis Report. https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html, 2022.

[21] The Linux Foundation. The State of Software Bill of Materials (SBOM) and Cybersecurity Readiness, 2022. 2025年1月28日に閲覧.

[22] The White House. Executive Order on Improving the Nation's Cybersecurity, 2021. 2025年1月28日に閲覧.

[23] Ying Wang, Bihuan Chen, Kaifeng Huang, Bowen Shi, Congying Xu, Xin Peng, Yijian Wu, and Yang Liu. An Empirical Study of Usages, Updates and Risks of Third-Party Libraries in Java Projects. In *Proceedings of the 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 35–45, 2020.

[24] Boming Xia, Tingting Bi, Zhenchang Xing, Qinghua Lu, and Liming Zhu. An Empirical Study on Software Bill of Materials: Where We Stand and the Road Ahead. In *Proceedings of the IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, pp. 2634–2646, 2023.

[25] 経済産業省 商務情報政策局サイバーセキュリティ課. ソフトウェア管理に向けた sbom（software bill of materials）の導入に関する手引. `https://www.meti.go.jp/press/2023/07/20230728004/20230728004.html`, 2023.

[26] 肥後芳樹. 自動テスト生成技術を利用した機能等価メソッドデータセットの構築. ソフトウェアエンジニアリングシンポジウム 2023 論文集, Vol. 2023, pp. 30–38, 08 2023.