一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report

OSSプロジェクトのメタデータを用いたソフトウェア選択支援ツールの試作

小林　亮太† 　松下　　誠† 　肥後　芳樹†

† 大阪大学大学院情報科学研究科 〒565–0871 大阪府吹田市山田丘 1–5
E-mail: †{ryt-kbys,matusita,higo}@ist.osaka-u.ac.jp

あらまし　オープンソースソフトウェア（OSS）の数は年々増加し，類似した機能を提供する OSS が複数存在するようになった．そのため利用者は，自身の要件に合った最適なソフトウェアを，多様な候補の中から選択する必要がある．しかし，これは選択の際に考慮する要因が多岐に渡るために困難を伴う．そこで本研究では，OSS のメタデータを用いてソフトウェア選択を支援することを目的としたツールを試作した．ユーザは，本ツールにより OSS の検索や機能・依存関係・ライセンスなどの把握ができ，容易に選択を行える．試作したツールの評価実験を実施し，試作ツールが既存ツールよりも効率的にソフトウェア選択を支援できることを確認した．

キーワード　OSS, ソフトウェア選択

1. まえがき

オープンソースソフトウェア（OSS）は，様々な分野や組織で広く利用されており [1]，その数は年々増加している．その結果，ある機能を果たすことができる，同じカテゴリーに属する OSS は複数存在している [2]．例えば，データベース管理を行う OSS には，MySQL や PostgreSQL などがあり，Web アプリケーションのバックエンドフレームワークには，Express や NestJS などがある．これらは，それぞれ特定の機能を果たせる類似した OSS であるが，特徴やライセンス，利用シーンなどに違いがある．従って，OSS の利用者や，ライブラリとして OSS を再利用して新たなソフトウェアを開発する開発者は，自身の要件に適した OSS を適切に選択する必要がある．

しかし，様々な理由から与えられた条件下における最適な OSS を選択することは難しい．第一に，OSS には膨大な選択肢が存在するためである [3]．同じ目的や機能を持つ OSS は数多く存在している．これは，利用者や開発者が自身の希望に沿った OSS を豊富な選択肢から選択することを可能にしている一方で，全体を把握した上でソフトウェアを選択することを困難にしている．第二に，ソフトウェアの選択要因は多岐に渡り [4]，様々な情報を参照する必要があるためである．OSS の情報は，ソースコード管理サイト GitHub や，公式ウェブサイト，ドキュメントなど様々な媒体で提供されており，その中から情報を取捨選択する必要がある．ライセンスや言語，コミュニティサポートなどの多数の情報を，複数の媒体から吟味するには時間的なコストがかかる [5]．これらの理由から，最適なソフトウェアの選択は非常に困難である．また，選択を誤った場合，開発プロセスや将来のメンテナンス活動に大きなコスト増を強いる可能性がある．

この問題を未然に防ぐため，多数の情報を効率的に吟味できるソフトウェア選択ツールを使用して最適な選択をするべきであるが，既存のツールではそれらは不十分である．例えば，オープンソース管理サイト OpenHub(注1)は，OSS のライセンスや機能などの情報を提供しているが，データが不完全なものが存在し，また依存関係の情報を確認できない．

そこで本研究では，多数の OSS のメタデータを用い，開発者が行うソフトウェア選択作業の支援を目的としたツールを試作した．本試作では，Ubuntu が提供している全てのパッケージを対象とし，OSS を収集した．まず，それらが提供している機能や開発者，ライセンス，依存関係などのプロジェクトのメタデータを収集した．その後，その情報を利用して類似している OSS プロジェクトを明らかにし，それらのデータを基にデータベースを構築した．最後に，ソフトウェアの選択要因を統一したフォーマットで提供する UI を開発した．ユーザは本ツールで，機能や名前，ライセンス，使用言語での OSS の検索を行うことができ，依存関係や類似プロジェクト，コミュニティの活動状況などを把握した上でソフトウェア選択を行える．また，開発したツールに関して評価実験を実施し，試作ツールがソフトウェア選択を効率的に支援できることを明らかにした．

以降，2. 節では関連研究や問題点について述べる．3. 節では試作ツールについて述べ，4. 節ではツールの評価実験について述べる．最後に 5. 節ではまとめと今後の課題を述べる．

2. 背　　景

本節では，ソフトウェアの類似性と，ソフトウェア選択および一般的なソフトウェア選択の手順について述べ，既存研究の限界と既存ツールの不十分な点について述べる．

2.1 ソフトウェアの類似性

ソフトウェア利用者や開発者にとって，類似しているソフ

(注1)：https://openhub.net/

トウェアを見つけることは，代替実装の探索や，関連するプロジェクトへの貢献に繋がる [6]．また，OSS プラットフォームにおける多数のリポジトリを研究する際にも必要不可欠である [7]．このように，類似したソフトウェアを特定する研究は，利用者や開発者，研究者にとって，興味深いトピックである．

ソフトウェアリポジトリ間の類似性を確立することに焦点を当てた研究は，使用するデータによって 3 種類に分類される．まず，MUDABlue [8] や CLAN [9] は，低レベルの情報であるリポジトリのソースコードのみを使用してリポジトリ間の類似性を見つける．次に，LibRec [10] や SimApp [11], RepoPal [12] は，高レベルの情報であるリポジトリのメタデータのみを使用して類似性を計算するアプローチである．最後に，CrossSim [13], [14] や Repo2Vec [7] は，ソースコードとメタデータなどの低レベルと高レベルの両方の情報を用いて類似性を発見する．特に，CrossSim は，プロジェクト名や開発者などをノードとし，依存関係など様々な関係をリンクに持つ OSS グラフから，SimRank アルゴリズム [15] によって，類似性を求める．OSS グラフとグラフ内のノードを参照するリストを与えるだけで，様々な言語のプロジェクトの類似性を計算できるため，汎用性が高い．

2.2 ソフトウェア選択

ある特定の機能を果たすことができる，同じカテゴリーに属する OSS は複数存在しているため，自身の要件に適した最適なソフトウェアを選択することが必要である [2]．誤ったソフトウェアの選択は，開発プロセスやメンテナンス活動に関して大幅なコスト増につながる可能性があり，選択は慎重に行わなければならない．

OSS の選択や評価に関する研究はこれまでにも多くの研究がなされている．OSS の選択要因を調査した研究 [4], [5], [16] では，OSS の選択要因が複雑であることや，組織や年代によって変化することが明らかになった．また，選択した OSS を評価するモデルも複数の研究で提案されている [2], [17], [18]．Adewumi らは，選択した OSS をビジネスや組織，技術環境などの全体的な要素を考慮して評価するモデルを提案し [2]，Tanzil らは，ライブラリに焦点を当て，ライブラリ選択プロセスのガイドラインの作成と選択ライブラリの評価モデルを提案した [18]．

2.3 ソフトウェア選択の手順

自身の要件に適したソフトウェアを選択する，一般的な手順は以下の通りである [18]．

（1） 要件定義
（2） 候補ソフトウェアのリストアップ
（3） 評価基準の設定
（4） 候補ソフトウェアの評価
（5） 評価基準に基づいた選択

（2）や（4）は，GiHub Advanced search(注2)や OpenHub での検索やその結果を利用して実現することが多い．

また，（3）で設定する評価基準としては，必須機能の適合度，ライセンス，言語，パフォーマンス，ソフトウェアの安全性などが挙げられる [18]．ソフトウェアの安全性は，依存関係の数や種類，コミュニティの活動状況などによって，総合的に判断する．これらの設定した基準に従って，各ソフトウェアを多角的かつ詳細に評価し，最適なソフトウェアを選択する．

2.4 問 題 点

2.2 節で述べた既存研究には，いくつかの問題が残されていると考える．調査研究 [4], [5], [16] では，ソフトウェアの主要な選択要因を明らかにしたが，ソフトウェアを選択する際に利用できる支援ツールの構築を行ったわけではない．評価モデル [2], [17], [18] は，ソフトウェア選択の度に毎回評価する必要がありコストが高いこと，評価指標を自由に調整できないことなどから，実際に使用されるまでには至っていないと考えられる．

また，2.3 節で紹介した GitHub Advanced search（GAS）と OpenHub では，ソフトウェア選択を行う上で問題があると考えられる．GAS の検索結果を表示する GitHub はソースコード管理サイトであるため，類似プロジェクトの情報は提供されておらず，キーワード検索以外に他の選択肢を探す手段は提供されていない．さらに，依存関係においても，直接的な依存関係を持つプロジェクトをリスト表示するのみであり，推移的依存や依存関係の複雑度を理解することは難しい．OpenHub はプロジェクト名や機能などでの検索を可能としているが，言語やライセンスなどの多数の情報を用いた検索は出来ない．そのため，粒度の粗い検索しか出来ず，詳細要件が異なるソフトウェアが多数表示されてしまう．また，一部データが不完全なものが存在し，それらのソフトウェアに関しては提供機能の説明や，ソースコードの安全性や言語情報などが得られない場合がある．さらに，依存関係の情報は提供されておらず，開発者が使用する際は，別のサイトで依存関係の情報を確認する必要がある．これらの問題は，効率的にソフトウェア選択に必要な情報を取得できず，最適なソフトウェアを選択することを困難にしている．

そこで本研究では，これらの既存研究の限界や既存ツールの問題点を解決する，ソフトウェアの検索や，選択要因を統一したフォーマットで提供することに焦点を当てたソフトウェア選択支援ツールを試作した．

3. 試作ツール

本節では，まず，2.4 節で述べた問題点を解決する手法のキーアイデアについて述べる．その後，試作ツールの概要について述べ，各コンポーネントの設計や機能を説明する．最後に，試作ツールの利用方法について述べる．

3.1 キーアイデア

ある特定の機能を果たすことができるソフトウェアを選択する場合，選択者は検索と各種要件の確認という 2 つの作業を行うと考える．2.4 節から，提案するソフトウェア選択支援ツールに必要と考えられる機能のキーアイデアは以下の通りである．

- 機能や名前だけではないオプションを用いた検索機能

(注2)：https://github.com/search/advanced

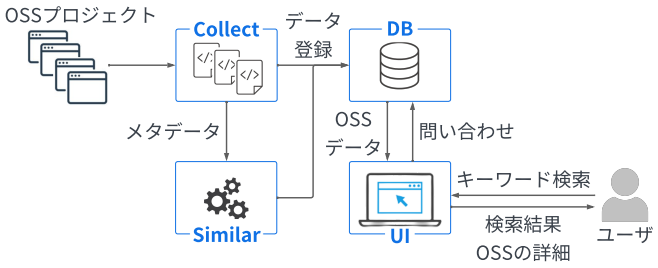

図 1　試作ツールの概要

- 機能説明やライセンス，言語などの基本的な要件の表示
- 依存関係に関する詳細情報の表示
- 他の選択肢の探索を容易にする類似プロジェクトの表示
- 安全性の評価基準となるコミュニティの活動状況の表示

これらのキーアイデアを実現することで，OpenHub での詳細要件が異なるソフトウェアが検索結果に表示される問題や，GitHub や OpenHub で確認できないより詳細な情報を提供できる．また，類似プロジェクトの情報を表示することで，効率的に他の選択肢を探すこともできる．これにより，ユーザはソフトウェアの選択要因を効率的に確認でき，容易に選択を行える．

3.2　概　　要

試作ツールは 4 つのコンポーネントで構築されている（図 1）．Collect コンポーネント（CC）は OSS プロジェクトからメタデータを収集する役割を持つ．Similar コンポーネント（SC）は CC が抽出したメタデータを利用して，各プロジェクト間の類似度を計算する．DB コンポーネント（DC）は CC で収集したメタデータと SC で計算した類似プロジェクトのデータを DB へ登録し，DB を動かす役割を担う．最後に，UI コンポーネント（UC）はユーザが利用できる UI を提供し，検索キーワードに一致する OSS プロジェクトを DC に問い合わせ，検索結果やプロジェクトの詳細データをユーザに表示する役割を持つ．

3.3　DB コンポーネント

DB コンポーネントは，CC で収集したメタデータと SC で求めた類似プロジェクトの情報を保持するデータベースである．データベースは MongoDB を採用した．3.1 節で述べた必要機能を提供するために，表 1 のフィールドをデータベースのスキーマとして定義した．ここで，表 1 の Package は，バイナリパッケージを提供している場合に，その機能や実行可能なアーキテクチャ名，実行に必要な依存関係を保持するためのフィールドである．また，Similars と Contributor は，Top5 の情報を保持する．

3.4　Collect コンポーネント

Collect コンポーネントは，表 1 に示す Similars 以外のデータを各 OSS プロジェクトから収集する．代表的な Linux のディストリビューションが提供している OSS プロジェクトから，これらのデータを抽出し，DC へ登録する．試作段階では，Ubuntu が提供する OSS プロジェクトを対象にして，パッケージのソースコードを取得し，表 1 に上げられる項目のメタデータを，ソースコード，control ファイル，copyright ファイル等の解析によって実現した．また，Commit や Contributor の情報はそのプロジェクトが使用しているバージョン管理システムから取得した．

表 1　データベースのフィールド

フィールド	説明
Name	OSS の名前
Distribution	ディストリビューション名
Section	OSS の分類
Description	提供機能の説明
Maintainers	開発者の名前とメールアドレス
License	ライセンス名
URL	ホームページやリポジトリなどのリンク
Language	書かれた言語の名前と割合
LOC	コード行数
Build-Depends	ビルドに必要な依存関係
Package	バイナリパッケージの情報
Similars	類似プロジェクト
Commit	コミット情報
Contributor	主な貢献者の情報

3.5　Similar コンポーネント

Similar コンポーネントは，CC で収集したメタデータから各プロジェクト間の類似度を計算する．計算には 2.1 節で紹介した CrossSim [13], [14] を使用し，OSS の名前，分類，開発者名，ホームページやリポジトリなどのリンク，ビルドに必要な依存関係の 5 つの情報を利用する．5 つの情報から，OSS グラフとグラフ内のノードを参照するリストを作成し，各プロジェクト間の類似度を計算する．その後，各プロジェクトに対して，計算した類似度が大きい 5 つを Similars フィールドへ登録する．

3.6　UI コンポーネント

UI コンポーネントは，ユーザが利用できる UI を提供し，検索キーワードに一致する OSS プロジェクトを DC に問い合わせ，検索結果やプロジェクトの詳細データをユーザに表示する．3.1 節で述べた必要機能を提供するために，オプションやカスタムスコアリングを利用した柔軟な検索，検索結果の一覧表示，プロジェクトの詳細表示を行える UI を作成した．

検索機能を実装した画面を図 2 に示す．検索機能では，フリーフォーマットのキーワードによって OSS を検索できる（赤枠部分）．検索時のオプションとして，ソフトウェア検索時に使用されることの多い言語とライセンスの 2 つのフィールドを設定した．言語は書かれた言語だけでなく，その OSS を使用できる言語でも検索が可能である（青枠部分）．検索はこれら 3 つの情報を用いて実施され，スコアリングによりスコアの高い順に結果が表示される．ここで，カスタムスコアリング機能により，ユーザは 3 つの情報の重要度を自由に変更でき，結果の表示順を変更できる（緑枠部分）．これらの機能により，オプションを用いた柔軟な検索を可能にしている．

一覧表示機能では，検索結果をスコア順に表示する．ここでは図 3 に示すように，機能の簡単な説明や，ソフトウェアの分類，書かれた言語，ライセンスといった基本的な要件のみを

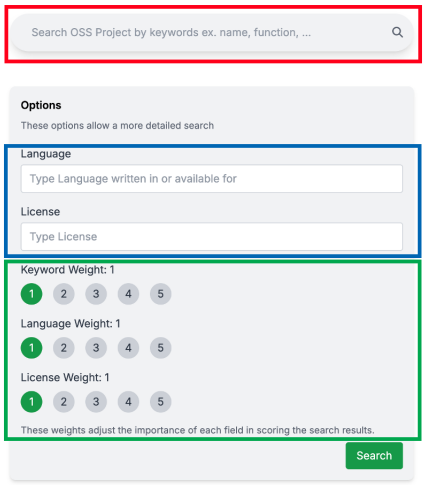

図 2　検 索 画 面

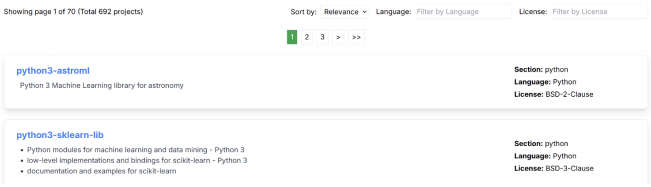

図 3　検索結果の一覧画面

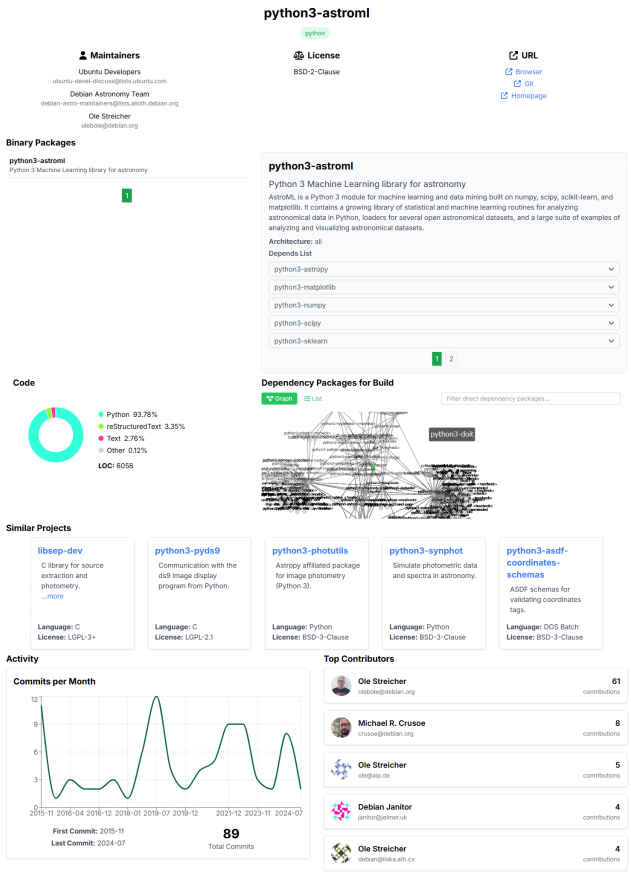

図 4　各 OSS の詳細画面

表示する．

表 1 の各種情報は，図 4 に示す詳細ページで表示する．詳細画面では，上部には OSS の名前や機能の説明などの基本的

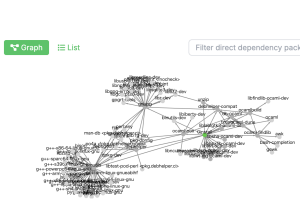

図 5　依存関係グラフ

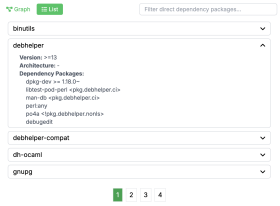

図 6　依存関係リスト

な要件を，中部にはコード構成と依存関係の情報，下部には類似プロジェクトやコミット，貢献者の情報を表示する．

依存関係の情報は，グラフとして表示する依存関係グラフ（図 5）とリストとして表示する依存関係リスト（図 6）の 2 つの方法で表示する．これらの依存関係は注目している OSS の直接的な依存関係と，その依存プロジェクトが依存しているプロジェクトを表示している．ここでグラフのノードは OSS プロジェクトであり，エッジは依存関係である．注目している OSS は緑色のノードで表現され，直接依存しているプロジェクトは濃い灰色，推移的依存のプロジェクトは薄い灰色で表現される．エッジは参照元から参照先に依存関係がある場合に接続される．また，依存関係リストはトグルを展開することで，依存プロジェクトのバージョン，アーキテクチャ，依存プロジェクトを確認できる．これら 2 つの表示により，依存関係の詳細情報を提供する．類似プロジェクトは，類似度が大きい上位 5 つを表示することで，他の選択肢の探索を容易にする．さらに，月毎のコミット数を折れ線グラフで表示し，代表的なプロジェクトの貢献者を最大 5 人まで表示することで，コミュニティの活動状況を表示する．

3.7　利 用 方 法

ユーザはキーワードによって OSS を検索でき，その検索結果や詳細データから，候補ソフトウェアのリストアップや評価基準に従った選択を行える．

まず，図 2 の検索画面で検索を行う．「machine learning」などのキーワードを入力し，言語やライセンスの希望があれば，「python」や「MIT GPL」と指定できる．言語やライセンスは半角スペースで区切ることで，複数の言語やライセンスを同時に指定し，OR 検索を実行できる．重要度を変更したいのであれば，カスタムスコアリング機能で重みを変更し，検索する．

検索結果は図 3 の一覧画面で表示される．ここでは，プロジェクトの名前，機能の簡単な説明，分類，書かれた言語，ライセンスを確認できる．また，ソートとフィルター機能も利用できる．ソート機能では関連度順のほかに，名前順に並び替えることができ，フィルター機能では，言語やライセンスを入力することで，検索結果のフィルタリングが可能である．ユーザは，一覧画面で基本的な要件を満たしているプロジェクトを発見し，候補ソフトウェアのリストアップができる．

図 4 の詳細画面では，プロジェクトの開発者名やホームページなどへのリンクを表示し，提供機能の説明（複数のバイナリパッケージを提供している場合，それぞれについての説明）を確認できる．また，依存関係グラフと依存関係リストにより，ビルドに必要な依存関係も確認できる．依存関係グラフ

は依存関係の数が多いほど密なグラフになるため，ユーザが依存関係全体の複雑度を直感的に理解することを手助けする．また，依存関係リストは依存プロジェクトの詳細情報を確認するのに適している．ユーザは，この詳細情報から自身の環境で依存関係の競合が発生しないかを理解するのに役立てることができる．さらに，詳細画面では表示される情報のフィルター機能も利用でき，効率的に特定の依存関係の情報に辿り着くことができる．これらの情報とコミットや貢献者の情報を利用することで，ユーザはプロジェクトの安全性を総合的に判断できる．

例えば，依存関係の数が少ない場合は，互換性の高いプロジェクトである可能性が高く，長期的にコミットがされているプロジェクトは，コミュニティの活動が歴史的かつ活発であり，バグ修正やメンテナンスが頻繁にされていると判断できる．また，貢献者の情報から，ユーザはどのような開発者がコミュニティに所属しているのかを理解でき，過去の経験などに基づいて今後の発展度合いや安全度を予想することに役立てることができる．さらに，何らかの要件を満たさない場合には，類似プロジェクトの情報を用いて他の選択肢を確認することも容易である．

これらの機能によって，試作ツールは，2.3 節で述べたソフトウェアを選択する一般的な手順のうち，選択者が個別に実施する（1）と（3）以外の作業を効率的に支援できると考える．（2）の候補ソフトウェアのリストアップは，オプションを用いた検索や検索結果の一覧画面の内容を確認することによって実現可能である．また，（3）で設定される主な評価項目を詳細画面で確認できるため，（4）の候補ソフトウェアの評価と（5）の評価基準に基づいた選択も容易である．

4. 評価実験

本節では，試作ツールがソフトウェア選択を効率的に支援できることを示す評価実験について述べる．比較対象として，GitHub Advanced Search（GAS），OpenHub，Ubuntu の公式リポジトリ(注3)の 3 つの既存ツールを使用して，ソフトウェアの検索を行い，ソフトウェアを探索した．Ubuntu の公式リポジトリを使用したのは，試作段階でのデータは Ubuntu のメタデータで構成されているからである．

4.1 実験デザイン

本評価実験は，2.3 節で述べたソフトウェアを選択する一般的な手順に則って行う．初めに，取り組む課題について述べ，その後，試作ツールと 3 つの既存ツールでの検索方法について述べる．最後に，評価に用いるメトリクスについて述べる．

4.1.1 課　　題

小規模から中規模程度の Web アプリケーションのバックエンドを構築するという課題を設定し，ソフトウェアの選択を行う．詳細な要件は表 2 に示す．スクリプトの柔軟性を重視した言語の指定や，商用利用することを想定したライセンスの指定など，実際に OSS を利用する際に考慮すると考えられる要件を反映した．

表 2　選択ソフトウェアの要件

項目	要件
機能	Web アプリケーションの バックエンドを構築することが可能
言語	Python が使用可能
ライセンス	MIT もしくは Apache2.0
パフォーマンス	速度は重視しないのであまり気にしない
ソフトウェアの 安全性	重要であり，特に以下の項目を重視 長期に渡って更新があり， 定期的にメンテナンスされていること 他の OSS との互換性のために， 依存関係が比較的少ないこと

4.1.2 検 索 方 法

表 2 の要件を満たす候補ソフトウェアを，3 つの既存ツールと試作ツールで検索する方法についてまとめたものを表 3 に示す．OpenHub に関しては，オプションを利用した検索を提供していないため，キーワードのみの検索を行う．

4.1.3 評価メトリクス

検索結果の上位 5 つについて表 2 の要件を満たしているかどうかを確認する．要件を満たしていれば 1 点，満たしていなければ 0 点であり，5 つの項目を 5 点満点で評価する: P_function, P_language, P_license, P_maintenance, P_depends．また，5 つの項目のうち，ツール内で確認できなかった要件を明らかにする．これにより，ソフトウェア選択の主な要因をツールがどの程度提供しているのかを評価する．さらに，要件全てを満たすプロジェクトが検索結果の上位何番目に現れるかを N_ans とし，どの程度早く目的のプロジェクトに辿り着けるのかを評価する．

4.2 結　　果

結果を表 4 に示す．また，ツール内で確認できなかった要件として，OpenHub は依存関係の数や種類を確認できず，Ubuntu はラインセンスやメンテナンス状況を確認できなかった．これらの情報に関しては，ソースコード管理に使用しているリポジトリの情報などを閲覧することで，明らかにした．GAS と試作ツールでは全ての要件をツール内で確認できた．

4.3 考　　察

表 4 の P_language と P_license の値が，OpenHub や Ubuntu は 5 ではないのに対して，試作ツールでは 5 となっている．これは，基本的な選択要因である言語とライセンスをオプションに含めることで，有用な選択肢のみを効率的に吟味できることを表している．また，P_fuction の値が既存ツールとほぼ同じ結果を示したことや N_ans が高い値を示したことから，試作ツールは既存ツールと同様の検索性能を持っていると考えられる．

試作ツールは，主要な選択要因を詳細画面によって提供している．そのため，OpenHub や Ubuntu のようにツール内で確認できない要件を他の媒体で明らかにするといった手間は最小限に抑えることができる．また，GAS は GitHub リポジトリでこれらの要件を確認できるが，ページの切り替えが必要であったり，得られる依存関係は直接的な依存関係のみであっ

(注3)：https://packages.ubuntu.com/

表3 検索方法

ツール	検索ワード	使用オプション: オプション値
GAS	web application framework	Working in this language: Python, With this license: MIT, Apache2.0
OpenHub	web application framework for python	-
Ubuntu	web application framework for python	検索対象: パッケージ説明, Distribution: すべて, セクション: すべて
試作ツール	web application framework	Language: Python, License: MIT Apache2.0

表4 結　　果

	GAS	OpenHub	Ubuntu	試作ツール
P_function	3	5	3	3
P_language	5	4	1	5
P_license	5	2	2	5
P_maintenance	4	5	4	5
P_depends	2	3	5	5
N_ans	2	25	23	3

たりすることから，試作ツールの方がより容易に詳細な情報を入手できることが分かる．さらに，GAS では類似プロジェクトの情報を提供していないため，機能要件は満たしているが，他の要件を満たしていない場合に，再度検索結果から探索することが必要となる．試作ツールでは類似プロジェクトの情報から容易に他の選択肢に辿り着くことができる．

これらのことから，試作ツールは既存ツールよりも効率的にソフトウェア選択を支援できるのではないかと考えられる．

5. まとめと今後の課題

本研究では，OSS プロジェクトのメタデータを用いてソフトウェア選択を支援するツールを試作した．試作段階では，Ubuntu が提供している全てのパッケージに含まれる OSS からメタデータを収集した．また，評価実験を実施し，試作ツールが既存ツールよりも効率的に支援できることを明らかにした．

今後の課題は，データの拡張と被験者実験の実施である．現在 DB に保持されているデータは Ubuntu のデータのみである．そのため，他の代表的な Linux のディストリビューションが提供している OSS からメタデータを収集し，データを増やす必要がある．また，被験者実験では，ツールの使用感や機能性を評価する．今回実施した評価実験では，課題要件を設定し，効率的に要件を吟味できることを示した．被験者実験では，ソフトウェアを実際に選択し，選択に要する時間やツールの情報が選択にどれだけ役立ったかを明らかにすることで，ツールが効率的にソフトウェア選択を支援できることを示したい．

謝辞

本研究は JSPS 科研費 JP24H00692, JP21K18302, JP21H04877, JP23K24823，JP22K11985 の助成を受けたものです．

文　　献

[1] B. Rossi, B. Russo, and G. Succi, “Adoption of free/libre open source software in public organizations: factors of impact,” Information Technology and People, vol.25, pp.156–187, June 2012.

[2] A. Adewumi，S. Misra，N. Omoregbe, and L. Fernandez-Sanz, “FOSSES: Framework for Open ‐ Source Software Evaluation and Selection, ” Software: Practice and Experience，vol.49，pp.780–812，Feb. 2019.

[3] 伊原彰紀，大平雅雄，“『オープンソースソフトウェア工学』シリーズ オープンソースソフトウェア工学,” コンピュータソフトウェア，vol.33，no.1，pp.28–40，2016.

[4] X. Li, S. Moreschini, Z. Zhang, and D. Taibi, “Exploring Factors and Metrics to Select Open Source Software Components for Integration: An Empirical Study,” Journal of Systems and Software, vol.188, p.111255, June 2022.

[5] V. Lenarduzzi, D. Taibi, D. Tosi, L. Lavazza, and S. Morasca, “Open Source Software Evaluation, Selection, and Adoption: A Systematic Literature Review,” 2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp.437–444, 2020.

[6] K.W. Nafi, B. Roy, C.K. Roy, and K.A. Schneider, “A Universal Cross Language Software Similarity Detector for Open Source Software Categorization,” Journal of Systems and Software, vol.162, p.110491, 2020.

[7] M.O.F. Rokon, P. Yan, R. Islam, and M. Faloutsos, “ Repo2Vec: A Comprehensive Embedding Approach for Determining Repository Similarity ,” 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp.355–365, 2021.

[8] S. Kawaguchi, P.K. Garg, M. Matsushita, and K. Inoue, “MUDABlue: An Automatic Categorization System for Open Source Repositories,” Journal of Systems and Software, vol.79, no.7, pp.939–953, 2006.

[9] C. McMillan, M. Grechanik, and D. Poshyvanyk, “Detecting Similar Software Applications,” 2012 34th International Conference on Software Engineering (ICSE), pp.364–374, 2012.

[10] F. Thung, D. Lo, and J. Lawall, “Automated Library Recommendation,” 2013 20th Working Conference on Reverse Engineering (WCRE), pp.182–191, 2013.

[11] N. Chen, S. Hoi, S. Li, and X. Xiao, “SimApp: A Framework for Detecting Similar Mobile Applications by Online Kernel Learning,” WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining, pp.305–314, Jan. 2015.

[12] Y. Zhang, D. Lo, P.S. Kochhar, X. Xia, Q. Li, and J. Sun, “Detecting Similar Repositories on GitHub,” 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), pp.13–23, 2017.

[13] P.T. Nguyen, J. Di Rocco, R. Rubei, and D. Di Ruscio, “CrossSim: Exploiting Mutual Relationships to Detect Similar OSS Projects,” 2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), pp.388–395, 2018.

[14] P. Nguyen, J. Rocco, R. Rubei, and D. Di Ruscio, “An automated approach to assess the similarity of GitHub repositories,” Software Quality Journal, vol.28, pp.595–631, 06 2020.

[15] G. Jeh and J. Widom, “SimRank: A Measure of Structural-Context Similarity,” Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.538–543, Aug. 2002.

[16] M. Sarrab and O.M.H. Rehman, “Empirical study of open source software selection for adoption, based on software quality characteristics,” Advances in Engineering Software, vol.69, pp.1–11, 2014.

[17] E. Bjarnason, P. Aberg, and N. binAli, “Software selection in large-scale software engineering: A model and criteria based on interactive rapid reviews,” Empirical Software Engineering, vol.28, pp.28–51, 2023.

[18] M.H. Tanzil，G. Uddin, and A. Barcomb, “ “How do people decide?”: A Model for Software Library Selection , ” 2024 IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)，pp.1–12，April 2024.