修士学位論文

題目

Ellecto: 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

指導教員
肥後 芳樹 教授

報告者
小林 亮太

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

Ellecto: 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

小林 亮太

内容梗概

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，これは選択肢が豊富であること，選択要因が多岐に渡ることから，非常に困難であり，選択作業を支援するツールにも不十分な点が存在すると考えられる．例えば，オープンソース管理サイト OpenHub は，OSS のライセンスや機能などの情報を提供しているが，依存関係の情報は提供していない．

そこで本研究では，ソフトウェア選択者が選択時に実施すると考えられる作業を効率的に行うために，言語や開発歴などのオプションを含めた名前検索に限らない柔軟なプロジェクト検索機能，ライセンスや言語などといったプロジェクト自体の基本的な要件の表示，プロジェクト間の依存関係の情報や開発コミュニティの活動状況の表示，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といった，ソフトウェア選択支援ツールに必要な機能のキーアイデアを提案する．また，提案したキーアイデアを基に，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ことを目的としたツール Ellecto を作成した．Ubuntu と Debian が提供する OSS プロジェクトを対象にメタデータを収集し，ツールを構築した．ユーザは，Ellecto 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．

さらに，Ellecto について被験者実験を実施した．実験には，Ellecto，OpenHub，GitHub Advanced search と GitHub の 3 つのツールを使用した．被験者はそれぞれのツールを使用し，3 つの課題で計 6 つのソフトウェアを選択した．実験の結果，被験者の 75%がソフトウェア選択を行う上で Ellecto を最も使用したいと回答した．また，選択ソフトウェアの正答率や回答に要した時間から，Ellecto がソフトウェア選択を効率的に支援でき，有用であることが分かった．

主な用語

OSS
ソフトウェア選択

目 次

1　はじめに

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 [20]，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 [1]．例えば，データベース管理を行う OSS には，MySQL や PostgreSQL，MariaDB[1] 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，利用状況などに違いがある．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 [27]．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 [11]，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，複数の媒体から探索し，評価するには時間的なコストがかかる [10]．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探索し，評価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だと考える．例えば，オープンソース管理サイト OpenHub[2]は，OSS のライセンスや機能などの情報を提供しているが，依存関係の情報は提供されていない．

そこで本研究では，ソフトウェア選択者が選択時に実施すると考えられる検索と各種要件の確認という 2 つの作業を効率的に行うための，ソフトウェア選択支援ツールに必要な機能のキーアイデアを述べる．また，提案したキーアイデアを基に，ソフトウェア選択作業の支援を目的としたツール Ellecto を作成した．まず，代表的な Linux のディストリビューションである Ubuntu と Debian 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．次に，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

[1] https://mariadb.org/
[2] https://openhub.net/

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 Ellecto で，機能や名前，ライセンス，使用言語などでの 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，Ellecto に関して評価実験を実施し，Ellecto がソフトウェア選択を効率的に支援でき，選択を行う上で有用であ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 節ではキーアイデアとそれを基に作成した Ellecto について述べる．4 節では Ellecto の評価実験について述べ，5 節では実施した評価実験の結果と考察について述べる．6 節では妥当性への脅威について述べる．最後に 7 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 [16]．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 [19]．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[9] や CLAN[15] 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[25] や SimApp[6]，RepoPal[26]，Collaborative Tagging[24] 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[17, 18] や Repo2Vec[19] 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 [8] 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 [1]．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 [10, 11, 22] 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 [1, 2, 5, 7, 14, 21, 23]．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 [1]，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 [23]．

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 [23].

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，OpenHub や GiHub Advanced search[3]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 [23]．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，OpenHub や GitHub リポジトリで提供されている情報，またはホームページなどの情報を基に，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 [10, 11, 22] 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 [1, 2, 5, 7, 14, 21, 23] 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 OpenHub と GitHub Advanced search（GAS）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.

OpenHub は図 1 のような検索画面で，プロジェクト名や機能などでのフリーワードによる検索を行える．しかし，オプションによって言語やライセンスなどを指定できず，多数の情報を用いた詳細な検索はできない．そのため，機能や名前のみで絞られた粒度の粗い検索結果が表示され，ライセンスなどの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．このことは，選択者により多くの項目を確認することを強い，時間的なコストがかかる．ま

[3] https://github.com/search/advanced

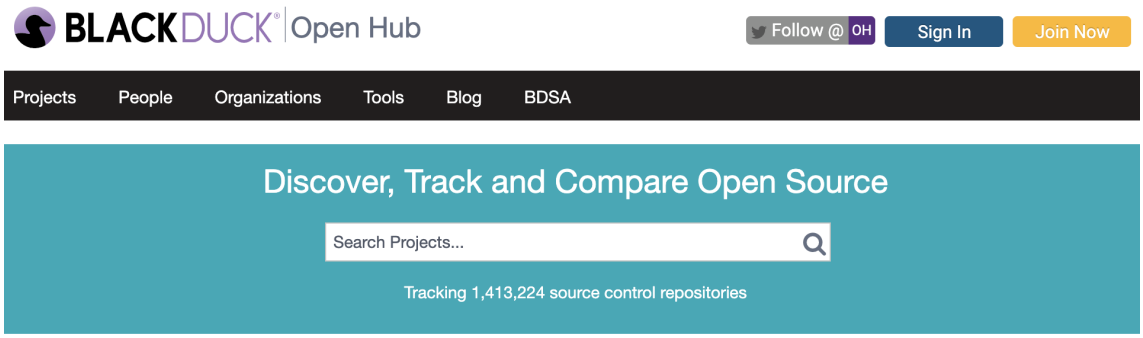

図 1: OpenHub の検索画面

た，OpenHub は依存関係の情報をソフトウェアの詳細情報として提供していない．そのため，選択時にソフトウェアの依存関係を考慮する選択者は，他のサイトで依存関係の情報を確認することが必要となり，多くの手間がかかってしまう．

GAS の検索結果はソースコード管理サイト GitHub で表示される．GitHub は，ソフトウェアの依存関係を図 2 のようなリスト形式で表示する．この表示では，直接的な依存関係と推移的な依存関係が区別されることなく表示されている．そのため，ある直接依存のプロジェクトがどのような推移的依存を持っているのか，どの程度推移的依存を持っているのか，といった依存関係の数や種類に関する詳細な情報を理解することは難しい．さらに，依存関係全体の複雑度を直感的に理解することも困難である．また，GitHub では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．これは，選択者が同様の機能を実現できる異なる特徴を持ったような他の選択肢を探すことを困難にしている．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と考える．

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 Ellecto を作成した．

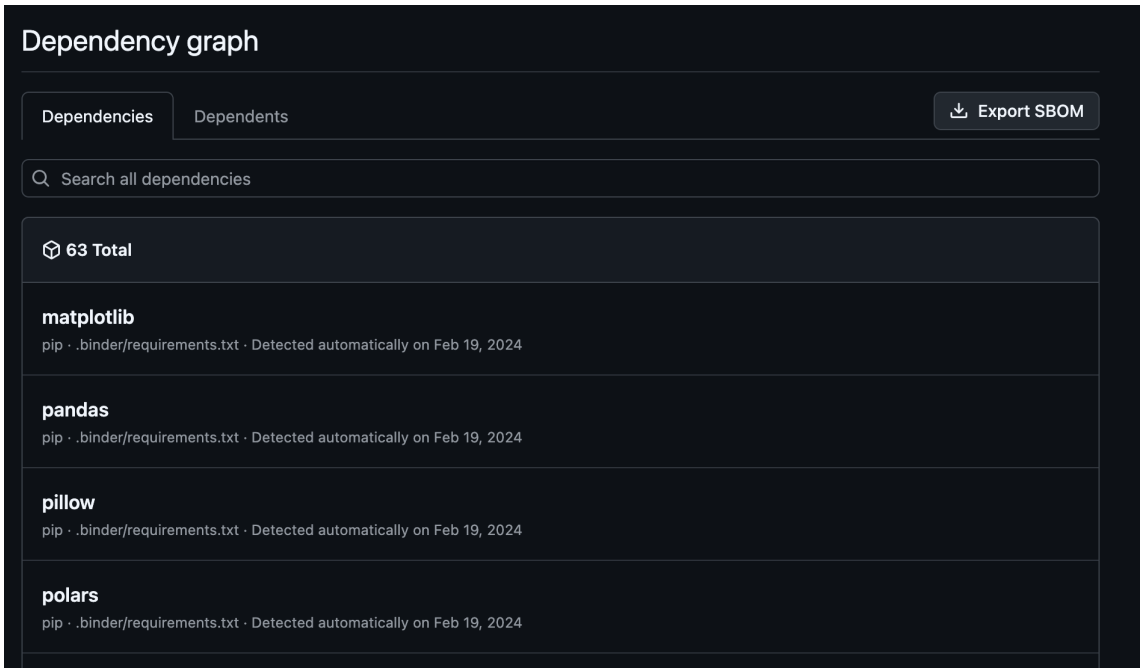

図 2: GitHub の依存関係表示画面

3 Ellecto

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，作成したツール Ellecto 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，Ellecto 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

言語やライセンスなどをオプションとして検索に含められるようにすることで，要件が一致するソフトウェアのみを検索結果に表示できる．これにより，OpenHub の詳細要件が異なるソフトウェアが多数表示される問題を解決し，ユーザは求めている要件のソフトウェアを見つけることが容易になる．

機能説明やライセンス，言語などの基本的な要件の表示

検索結果を一覧表示する画面にこれらの基本的な要件を表示することで，全体を把握

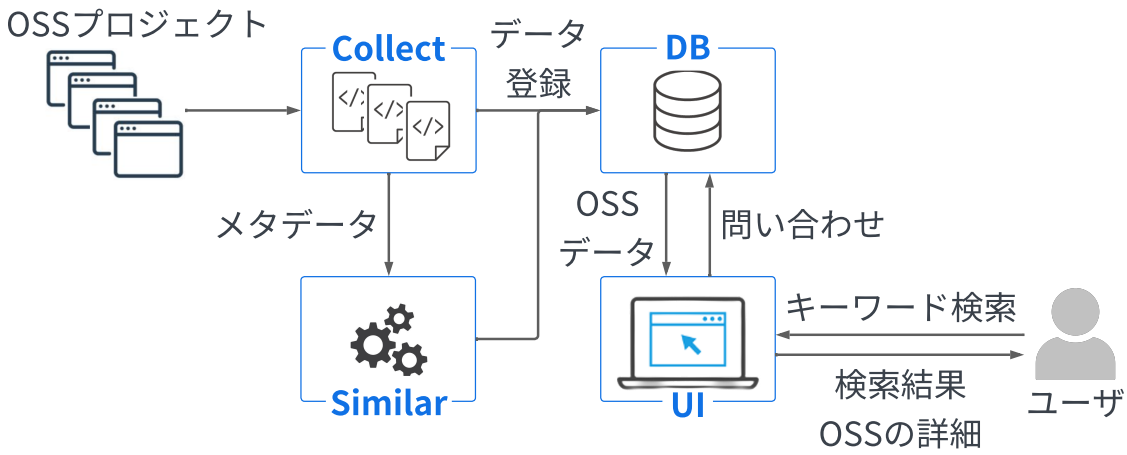

図 3: Ellecto の概要

しながら候補となるソフトウェアを検索結果から探すことができる．

詳細な依存関係の情報やコミュニティの活動状況の表示

直接的な依存関係のプロジェクトが持つ推移的依存関係の数や種類を表示することや，依存関係の全体的な複雑度を表示することで，ユーザはソフトウェアの詳細な依存関係の情報を理解できる．また，コミュニティの活動状況を表示することで，プロジェクトの活動歴や開発の活発度も理解できる．これにより，GitHub や OpenHub で確認できないより詳細な依存関係の情報を提供し，コミュニティの活動情報なども含めて多角的な視点からソフトウェアの安全性を評価できる．

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

機能的には充分であるが，詳細要件が満たされていないソフトウェアがあった場合に，その類似プロジェクトを表示することで，代替の選択肢を容易に発見できる．

3.2 概要

Ellecto は 4 つのコンポーネントで構築されている（図 3）．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．

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
Depends	実行時に必要な依存関係
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
FirstCommitDate	最初のコミット日
LastCommitDate	最新のコミット日
Commit	月ごとのコミット数の情報
Similars	類似プロジェクト
Contributor	主な貢献者の情報

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[4]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．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

[4] https://www.mongodb.com/

らのデータを抽出し，DC へ登録する．

表 1 に示すメタデータを収集する方法は，以下の通りである．

1. 対象ディストリビューションの Docker 環境を構築
2. パッケージリストを取得
3. 取得した各パッケージのソースコードを入手
4. メタデータを収集し，表 1 のフィールドに合うように正規化

Ellecto では，Ubuntu と Debian が提供する OSS プロジェクトを対象にして，上記の手順に従いメタデータを収集した．(2) は `apt list 2 > /dev/null > packages.txt` を Docker コンテナ内で実行することで実現した．また, (3) は (2) で取得した各パッケージ名に対して，`apt source` コマンドを実行することで，ソースコードを入手した．(4) は取得したソースコードから主に control ファイルを解析することによって実現した．License は copyright ファイルを解析し，Language や LOC はソースコードに対して `cloc` コマンド[5]を実行することで，データを収集した．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の API を使用することで取得した．

これらの方法で Ubuntu と Debian が提供する OSS プロジェクトを対象にしてデータを収集した結果，78001 個の重複のない OSS プロジェクトのメタデータを収集できた．収集したメタデータは DC で動かされている DB へ登録した．

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[17, 18] 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．まず，5 つの情報から，CrossSim の入力として必要な，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た．その後，CrossSim を用いて各プロジェクト間の類似度を計算し，最後に，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 DB の Similars フィールドへ登録した．

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

[5] `https://github.com/AlDanial/cloc`

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 4 に示す．検索機能では，OSS の名前や機能など，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，以下の 5 つのオプションを用いて検索が可能である（青枠部分）．

- 言語オプション

  このオプションにより，OSS が書かれた言語，または，使用できる言語を指定し，検索を行える．

- ライセンスオプション

  このオプションにより，OSS のライセンスを指定し，検索を行える．

- 開発歴オプション

  このオプションにより，OSS の開発歴の長さを指定し，検索を行える．開発歴の長さは年単位で指定できる．

- 最終更新日オプション

  このオプションにより，OSS の最終更新日を指定し，検索を行える．最終更新日は，最新のコミット日から x 年 y ヶ月以内のように指定できる．

- 依存関係オプション

  このオプションにより，OSS の実行時の依存関係の最大数を指定し，検索を行える．

検索はキーワードと最大 5 つのオプション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．各検索条件のスコアは，検索キーワードや言語，ライセンスの値は正規表現に変換し，それらと対象のフィールドの値が一致したものをカウントすることで計算される．検索キーワードは，Name，Section，Description フィールドの値を，言語は Language と Description フィールドの値を，ライセンスは License フィールドの値を参照する．また，開発歴オプションは検索日時から FirstCommitDate の値を引いた最初のコミットからの期間を参照し，最終更新日は検索日時から LastCommitDate の値を引いた最新のコミットからの期間を参照する．さらに，依存関係は実行時の直接的な依存関係の数を参照する．与えられた検索条件を AND 検索で検索し，検索条件を満たす OSS のみを検索結果に表示する．ここで，カスタムスコアリング機能により，ユーザはキーワードと 5 つのオプションの重要度を自由に変更でき，自身が求めている OSS を検索結果のより上位

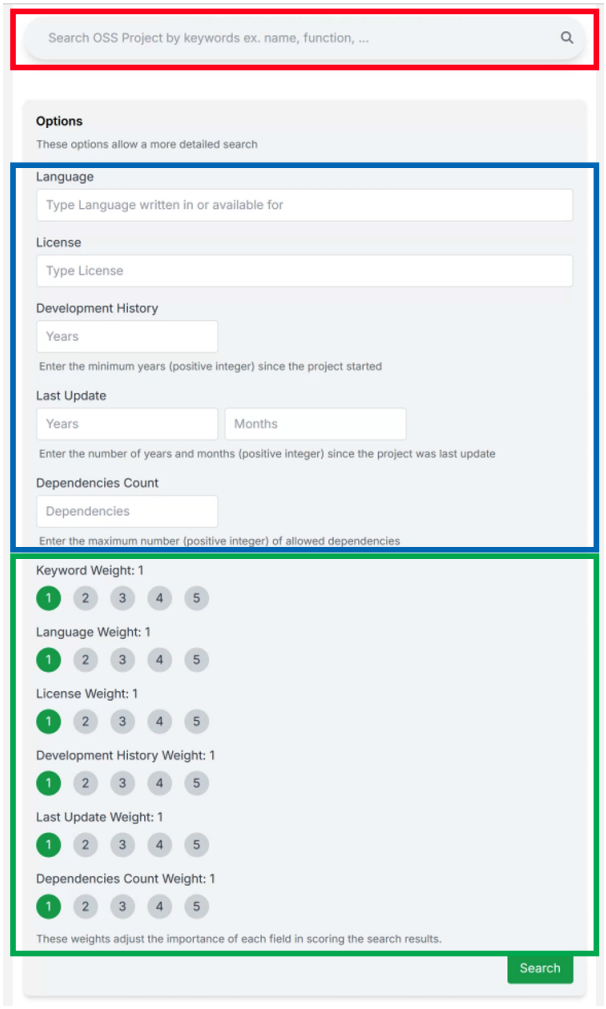

図 4: 検索画面

に表示できる（緑枠部分）．設定された重みを用いて，検索結果のスコアは以下の式で計算される．

$$\begin{aligned}\text{score} = {} & \text{keyword\_weight} \cdot \text{keyword\_score} \\ & + \text{language\_weight} \cdot \text{language\_score} \\ & + \text{license\_weight} \cdot \text{license\_score} \\ & + \text{devhistory\_weight} \cdot \text{devhistory\_score} \\ & + \text{lastupdate\_weight} \cdot \text{lastupdate\_score} \\ & + \text{depends\_weight} \cdot \text{depends\_score}\end{aligned}$$

このスコアを比較し，スコアの高い順に検索結果を表示する．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．

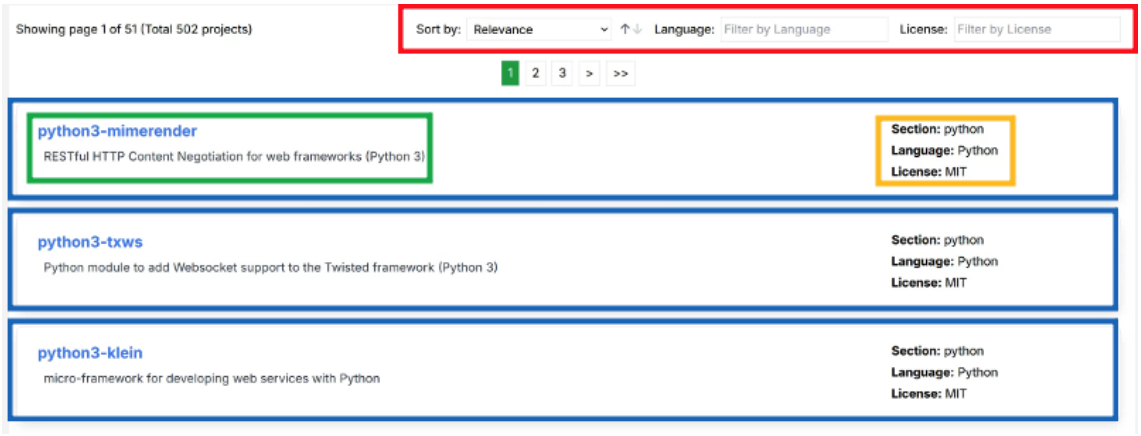

図 5: 検索結果の一覧画面

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 5 に示すように，検索条件に一致する OSS の名前や機能の簡単な説明など，基本的な要件のみを表示する．各 OSS の情報は，図 5 の青枠部分の UI で表示され，左側（緑枠部分）に OSS の名前と機能の簡単な説明を表示し，右側（橙色部分）に，OSS の分類，書かれた言語，ライセンスを表示する．また，ソートとフィルター機能を使用できる（赤枠部分）．ソート機能ではスコア順で並べた関連度順のほかに，名前順，開発歴順，更新日順，依存関係の数順に並び替えられる．矢印ボタンを押すことで，昇順と降順を入れ替えることもできる．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．

表 1 の各種情報は，図 6 に示す詳細ページで表示する．詳細画面は 5 つの画面で構成される．赤枠部分には，以下に示す 8 個の情報を表示する．

- OSS の名前
- OSS の分類
- プロジェクトの人気度指標

  プロジェクトが使用しているソースコード管理サイトのスター数とフォーク数を表示する
- 開発者の情報

  プロジェクトの開発者名と E-mail アドレスを表示する
- ライセンス
- 各種 URL

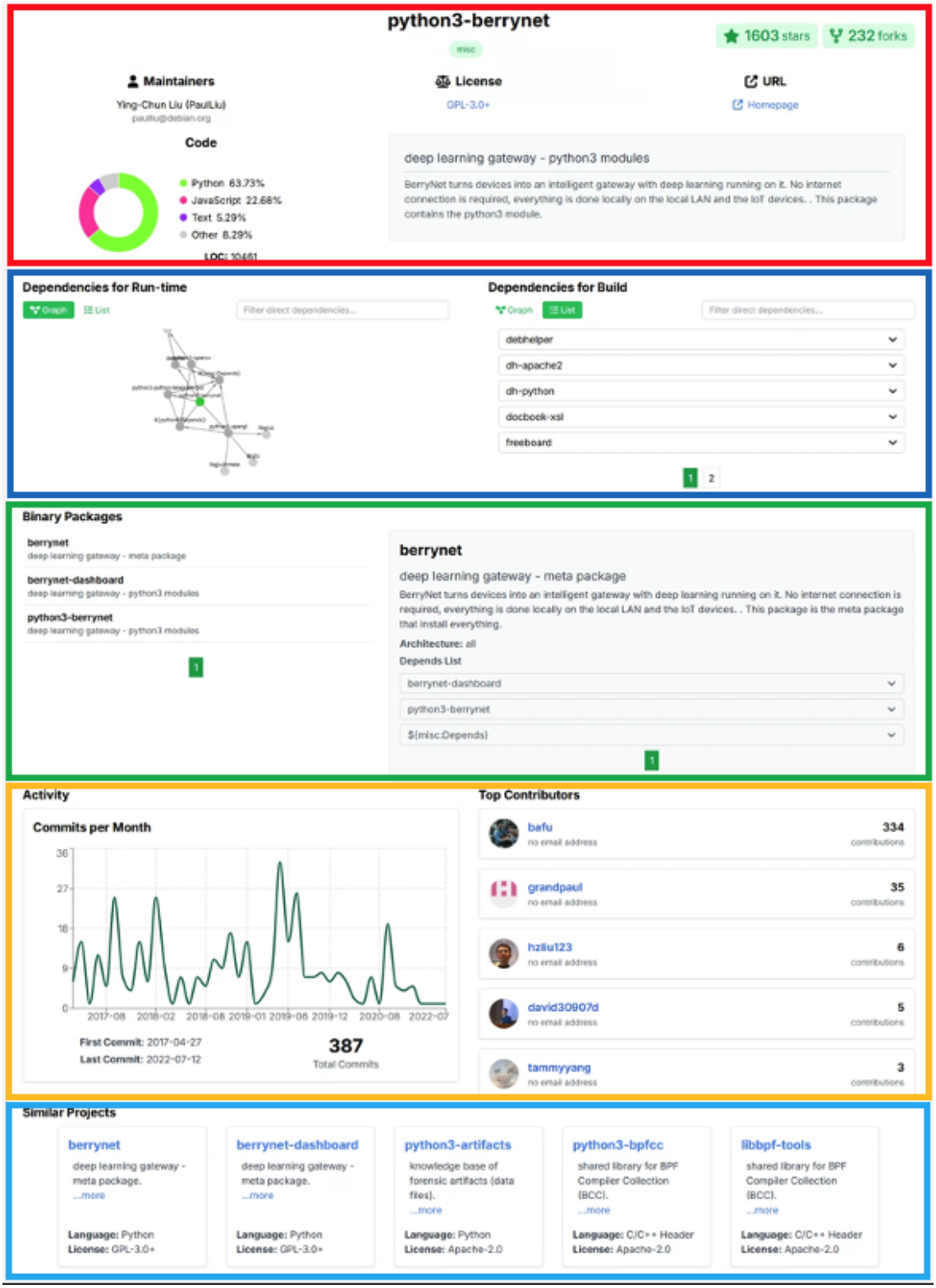

図 6: 各 OSS の詳細画面

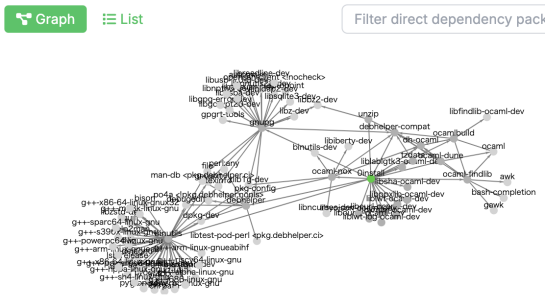

図 7: 依存関係グラフ

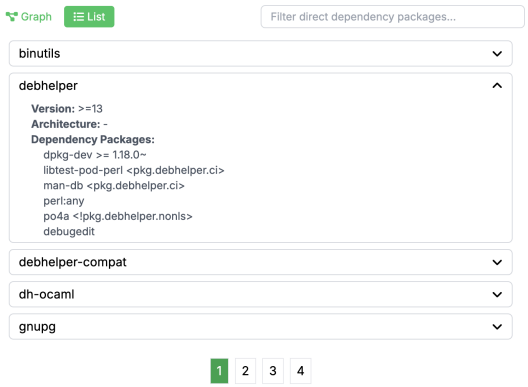

図 8: 依存関係リスト

プロジェクトのホームページ URL や，ソースコード管理サイトの URL などを表示する．

- コード構成

  プロジェクトのソースコードにおける各言語の割合と LOC を表示する．割合を表す円グラフは，上位 3 つのコードの割合と種類を表示し，それ以下の言語は Other としてまとめて表示する．

- 詳細な機能説明

青枠部分には，依存関係の情報を表示する．左側に実行時の依存関係を表示し，右側にビルド時の依存関係を表示する．それぞれの依存関係の情報は，グラフとして表示する依存関係グラフ（図 7）とリストとして表示する依存関係リスト（図 8）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．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．

緑枠部分には，バイナリパッケージの情報を表示する．左側に各バイナリパッケージをリスト表示し，右側にバイナリパッケージの名前や機能の詳細な説明，アーキテクチャ，実行時の直接的な依存関係を表示する．

橙色部分には，コミュニティの活動状況を表示する．左側に月毎のコミット数を表した折れ線グラフを表示し，右側に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．

水色部分には，類似した機能を持つプロジェクトを表示する．ここでは，SC で求めた類似度が大きい上位 5 つのプロジェクトの名前，機能の簡単な説明，書かれた言語，ライセンスを表示する．

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 4 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．また，開発歴や最終更新日，依存関係のオプションを使用することで，より詳細な検索を行える．開発歴は，例えば「10」とすると，少なくとも 10 年以上開発歴がある OSS で絞り込みができ，最終更新日は「1 年 6ヶ月」とすると，最終更新日が 1 年 6ヶ月以内の OSS を検索できる．依存関係の最大数を指定する場合は,「10」とすると，実行時の直接的な依存関係の数が 10 個以内の OSS のみを検索できる．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それぞれの検索条件の重みを変更し，検索する．

検索結果は図 5 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を使用することで，検索結果の並び替えや絞り込みもできる．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．詳細情報を確認したい場合は，プロジェクトの名前をクリックすることで詳細画面に遷移できる．

図 6 の詳細画面では，提供機能の説明やプロジェクトの開発者名，採用ライセンスなどを確認できる．また，依存関係グラフと依存関係リストにより，実行時に必要な依存関係とビルドに必要な依存関係を確認できる．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，Ellecto 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．

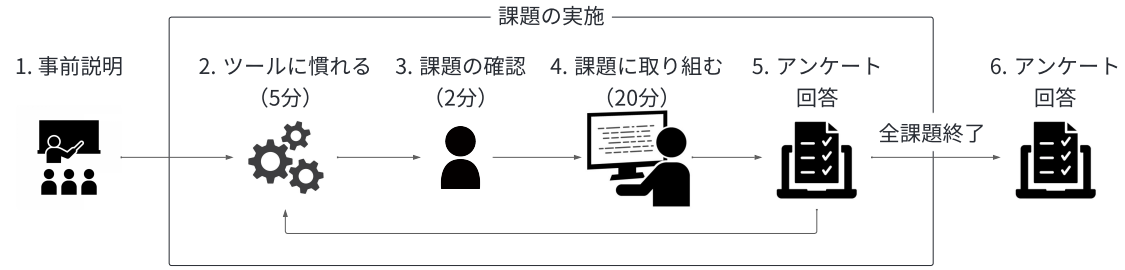

図 9: 実験手順

4　評価実験

本節では，Ellecto がソフトウェア選択を効率的に支援できることを示すために実施した被験者実験について述べる．まず，実験の概要について述べる．その後，被験者が取り組む課題について述べ，課題と回答に使用するツールのグループ分けについて説明する．最後に，評価指標について述べる．

4.1　概要

Ellecto がソフトウェア選択を効率的に支援できることを示すために被験者実験を実施した．実験では，被験者に選択ソフトウェアの要件をまとめた 3 つの課題を提示した．被験者はツールを使用してソフトウェアを検索し，要件を満たすソフトウェアを選択した．使用したツールは，Ellecto，OpenHub，GitHub Advanced search（GAS）と GitHub の 3 つである．以降，本節では実験手順，使用ツールごとの回答方法，課題取り組み時のルールについて説明する．

4.1.1　実験手順

実験は図 9 に示す手順に従って実施した．まず，事前説明として，実験の趣旨，使用ツールごとの回答方法（4.1.2 節），課題取り組み時のルール（4.1.3 節）を説明した．その後，被験者は 2 から 5 の 4 つのステップで課題に取り組んだ．まず，ツールによる熟練度の違いを緩和するため，課題回答に使用するツールに慣れる時間を設けた．ここでは，使用するツールを自由に操作でき，被験者はツールでの検索方法や操作方法，表示情報などを確認した．その後，課題を確認する時間を取った後，課題要件を満たすソフトウェアを最大 20 分で選択してもらった．最後に，被験者は使用したツールに関するアンケートに回答した．これらの一連の流れを 3 つの課題で実施し，全課題終了後には別のアンケートに回答した．また，被験者は 3 つの課題で別々のツールを使用した．被験者と使用ツールのグループ分けについ

	回答欄1	回答欄2	回答欄3	
データベースのクライアントツール				
YAMLファイルを操作するツール				
	x分y秒			
選択にかかった時間				

図 10: 回答シート

図 11: OpenHub の回答項目

ては，4.3 節で述べる．

4.1.2 使用ツールごとの回答方法

被験者は，図 10 に示す Excel シートを使用し，回答を記述した．回答欄に記入する内容は，使用ツールごとに指定した．Ellecto を使用した場合は，図 6 の詳細画面に表示されているプロジェクト名を，OpenHub の場合は図 11 赤枠部分の詳細画面のリンクを，GAS & GitHub の場合は図 12 赤枠部分のリポジトリのリンクを記入するよう指示した． 複数のソフトウェアで要求機能を実現するといった回答も可能であり，その場合，回答欄 2 や 3 を使用して必要なソフトウェア全てを記入するよう促した．

4.1.3 課題取り組み時のルール

被験者は，課題取り組み時に，以下の基準に従って使用ツール以外の外部サイトを使用，閲覧できることとした．また，それぞれのツールを使用して得られる情報からソフトウェア選択を行ったことを保証するため，以下の使用禁止にあげられるサイトの利用を明示的に禁じた．

- 使用，閲覧可能

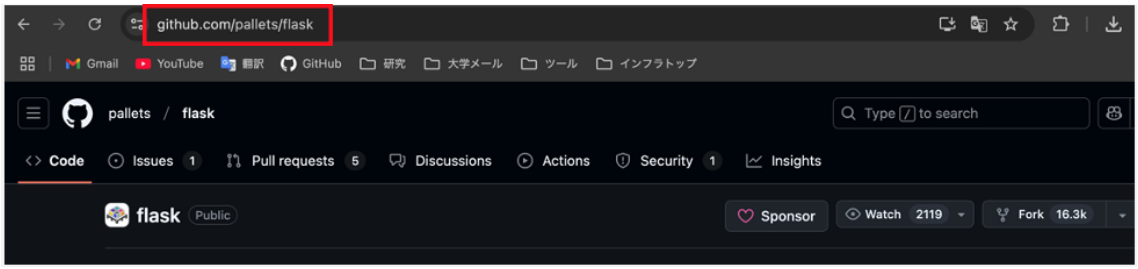

図 12: GAS & GitHub の回答項目

  - DeepL[6]や Google 翻訳[7]などの翻訳サイト
  - 各ツール内で記載されている情報から辿れるホームページやソースコード管理サイトなどで表示されている情報，およびそこからさらに辿れるサイトの情報

- 使用禁止
  - Google[8]などの検索エンジン
  - ChatGPT[9]などの生成 AI

4.2 課題

本実験では，選択ソフトウェアの要件をまとめた 3 つの課題を提示した．これらの課題は実際のプロジェクトの要件 [3, 4, 13] を基に作成した．被験者実験に使用した課題の全文は付録 A に示す．課題は，要件が緩いものや厳しいものなど様々な要件を設定した．

4.3 課題と使用ツールのグループ分け

本実験には，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に所属する修士の学生 8 名と学部生 4 名の計 12 名が参加した．表 2 に示すように，課題と使用ツールの組み合わせで被験者を 3 つのグループに分割した．また，ツールの使用順による結果の偏りを減らすため，課題の実施順は被験者ごとに変更した．

4.4 評価指標

実験の評価指標には，正答率，回答に要した時間，アンケートの回答結果を用いた．

[6] https://www.deepl.com
[7] https://translate.google.com
[8] https://www.google.co.jp/
[9] https://chatgpt.com/

表 2: 課題と使用ツールによるグループ分け

グループ	課題 1 使用ツール	課題 2 使用ツール	課題 3 使用ツール
A	OpenHub	GAS & GitHub	Ellecto
B	GAS & GitHub	Ellecto	OpenHub
C	Ellecto	OpenHub	GAS & GitHub

正答率は，選択したソフトウェアで要求機能を実現できるかを確認し，回答数に対して実現できた回答数の割合で表される．各課題，各ツールごとに正答率を計算した．

回答に要した時間は，課題を開始してから被験者が回答終了と判断するまでの時間である．タイマーを使用することで，被験者自身が回答終了までの時間を計測した．課題の実施時間は，実験手順に示すように最大 20 分である．

また，アンケートは以下の 2 種類のアンケートを実施した．

- 各課題終了後のアンケート

  このアンケートでは，被験者は直前に取り組んだ課題で使用したツールに関して，以下の 5 項目をリッカート尺度 [12] を用いた 5 段階評価で回答した．

  - 検索機能の使いやすさ
  - 検索オプションの有用性
  - 検索結果の画面や詳細画面での表示情報の網羅性
  - 検索結果の画面や詳細画面での表示情報の有用性
  - ソフトウェア選択を行う上での使用ツールの総合評価

- 全課題終了後のアンケート

  このアンケートでは，被験者はソフトウェア選択を行う上で使用したいと思う順に 3 つのツールを順位付けした．また，Ellecto の良かった点や改善点を自由記述形式で回答した．

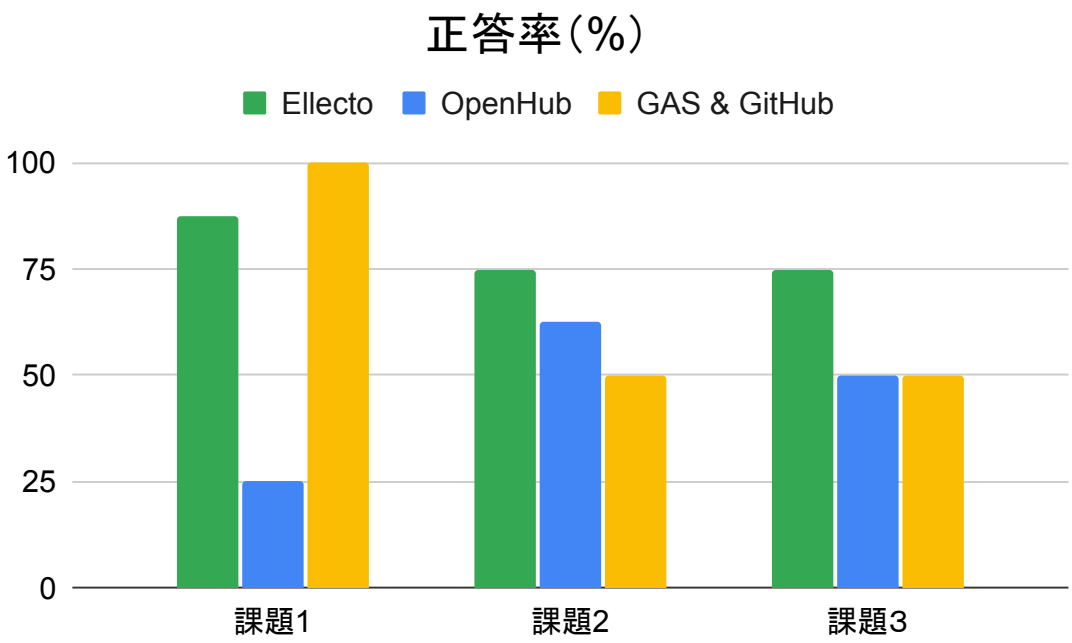

図 13: 各課題での使用ツールごとの正答率

表 3: 使用ツールごとの平均正答率

	Ellecto	OpenHub	GAS & GitHub
平均正答率	**79.2%**	45.8%	66.7%

5 実験結果と考察

本節では，4 節で述べた被験者実験を実施した結果と，その結果に基づく考察について述べる．

5.1 正答率

被験者は，付録 A に示す 3 つの課題で計 6 つのソフトウェアを選択した．選択したソフトウェアで要求機能を実現できるか，およびその他の条件を満たしているかを手作業で確認し，各課題，使用ツールごとに正答率を計算した結果を図 13 に示す．また，使用ツールごとの平均正答率（小数第 2 位で四捨五入した値）を表 3 に示す．

図 13 の結果から，課題 1 では GAS & GitHub が最も正答率が高く，課題 2 および 3 では Ellecto が最も正答率が高かったことが分かる．課題 1 は，付録 A.1 に示すように，機能要件以外の要件が無いソフトウェアを選択する課題があり，ここでの正答数が結果に影響し

表 4: 使用ツールごとの平均正答率

回答時間（中央値）	Ellecto	OpenHub	GAS & GitHub
課題 1	**9 分 18 秒**	19 分 57 秒	15 分 24 秒
課題 2	**13 分 30 秒**	18 分 53 秒	19 分 28 秒
課題 3	19 分 21 秒	**18 分 55 秒**	20 分 00 秒

た．このような要件が緩く，どのようなソフトウェアでも使用候補となる場合は，GitHub のような使い慣れたツールでソフトウェアを選択した方が容易である可能性がある．しかし，課題 2 および課題 3 では Ellecto を使用したときの正答率が最も高かった．これらの課題では，ライセンスや言語，開発履歴など複数の要件を考慮する必要があった．このような場合に Ellecto は効果的であり，3.1 節で述べた機能や名前だけでないオプションを用いた検索機能のキーアイデアが有効であったことが分かる．実際，他のツールでの誤答を分析すると，機能は実現できるが，Python で使用できないソフトウェアを選択したり，10 年以上の開発履歴がないソフトウェアを選択したりしていた．Ellecto では検索オプションで要件を満たすソフトウェアのみを検索結果に表示できるため，被験者は機能を実現できるかを調べるだけでよく，効率的にソフトウェアの選択を行えたと考えられる．

また，表 3 の結果から，使用ツールごとの平均正答率は Ellecto が最も高かった．このことから，Ellecto は従来のツールよりも正確にソフトウェアの選択を行えるツールであることが分かる．

これらのことから，Ellecto は複雑な要件になることが多いソフトウェアの選択要因を考慮でき，検索オプションの絞り込みによって効率的かつ正確にソフトウェアを選択できるツールであると考えられる．

5.2 回答に要した時間

各課題，使用ツールごとの回答に要した時間を計測した結果（中央値）を表 4 に示す．

表 4 の結果から，課題 1 および 2 では Ellecto を使用したときの回答時間が最も短く，課題 3 では OpenHub を使用したときの回答時間が最も短いことが分かる．課題 1 および 2 で Ellecto がこのような結果を示した理由として以下のことが考えられる．

- 検索オプションで絞り込んだ検索結果からソフトウェアを探索できたこと
- ソフトウェアの情報が詳細画面の一画面にまとめられていたこと
- 類似しているソフトウェアの情報から他の選択肢を探すことが容易であったこと

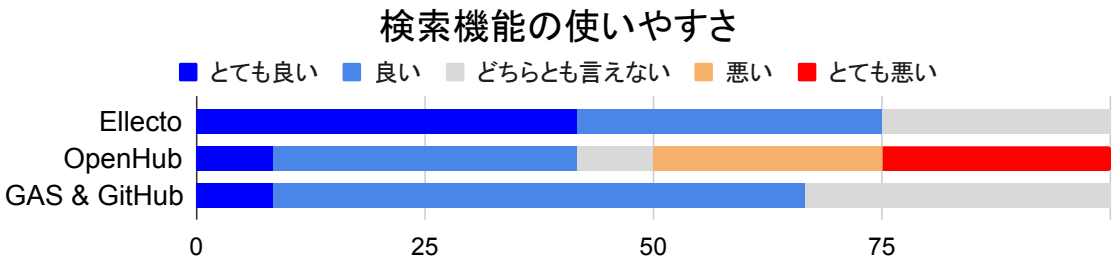

図 14: 検索機能の使いやすさのアンケート結果

これらは 3.1 節でのキーアイデアの内，オプションを用いた検索機能と類似プロジェクトの表示が有効的に働いた結果であると考えられる．一方で，課題 3 では OpenHub を使用したときの回答時間が最も短かった．これは，課題 3 の機能要件は複雑であったため，Ellecto でそれらを入力するのに時間を要したことが原因ではないかと考えられる．また，OpenHub では付録 A.3 に示すような機能を実現できるソフトウェアが少なく，検索結果は数件のみだった．Ellecto では多様なソフトウェアが検索結果に表示されており，その差分，比較や評価に時間を要したことも原因であると考えられる．しかし，図 13 の結果からも分かるように，課題 3 での正答率は Ellecto が最も高い．これは被験者が Ellecto の詳細画面で依存関係の情報を把握できたことや，検索オプションによって結果の絞り込みができたことでそれぞれのソフトウェアを正しく評価できた結果であると考える．OpenHub と Ellecto の中央値の差は 30 秒程度であり，正答率の結果からこの時間差は大きく影響しないのではないだろうか．

これらのことから，Ellecto はソフトウェア選択を効率的に支援できるツールであることが分かる．既存ツールを使用するよりも Ellecto を使用する方が，ソフトウェア選択に要する時間を短くでき，より正確な選択ができる．

5.3 アンケート

5.3.1 各課題終了後のアンケート

図 9 に示すように，被験者は各課題終了後に使用したツールに関してのアンケートに回答した．アンケート結果を図 14 から図 18 に示す． また，アンケートの項目ごとに，各ツールの回答の平均値を計算したもの（小数第 3 位で四捨五入した値）を表 5 に示す．

図 14 から図 18 の結果から，全ての項目において，Ellecto は肯定的な回答（とても良いまたは良い）をされた割合が同じか高かったことが分かる．また，全ての項目において，とても良いと回答された割合は最大であり，他のツールと比較して少なくとも 2 倍以上，最大で約 8 倍の差があった．さらに，全ての項目において，とても悪いと回答した被験者はおら

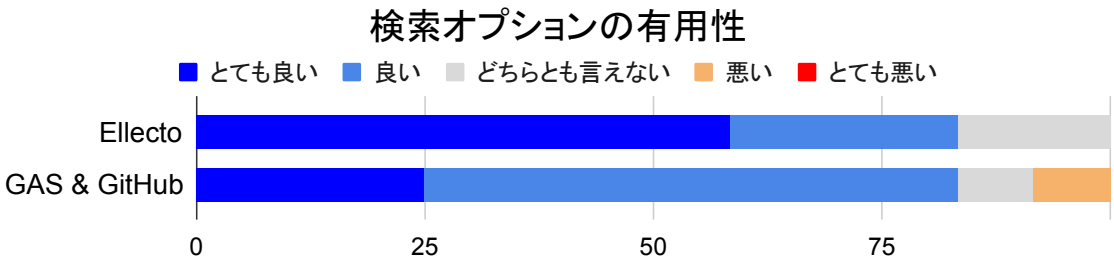

図 15: 検索オプションの有用性のアンケート結果

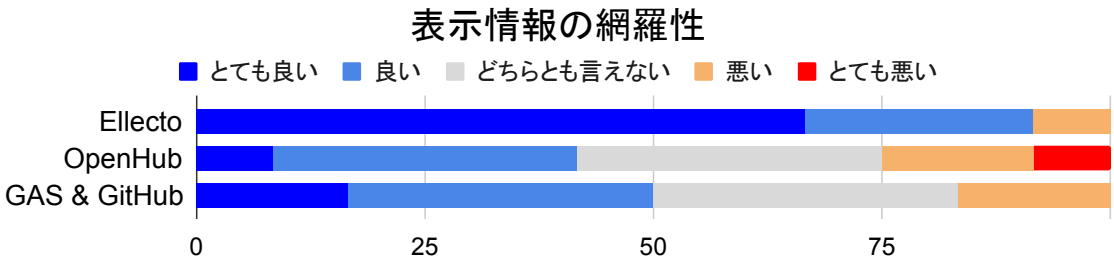

図 16: 検索結果の画面や詳細画面での表示情報の網羅性のアンケート結果

ず，悪いと回答した被験者は表示情報の網羅性における 1 人のみであった．また，表 5 の結果から，全ての項目において，Ellecto は回答の平均値が高かった．これらのことから，被験者は Ellecto が他のツールと比較して使いやすく，有用であったと判断したことが分かる．

図 14 と図 15 の結果が示すように，多くの被験者は Ellecto の検索機能が使いやすく，オプションが有用であったと回答した．これにより，オプションを用いた検索機能のキーアイデアを実装したことが，ソフトウェア選択に有益だったことが分かる．

図 16 と図 17 の結果が示すように，多くの被験者は Ellecto で表示されている情報は網羅的であり有用であると回答した．これは，3.3 節で述べた必要機能を提供するために定義したデータベースのフィールド（表 1）を基にメタデータを収集したことが有用であったことを表しており，設計の妥当性を示している．一方で，表示情報の網羅性において悪いと回答した被験者に回答理由を調査したところ，検索結果の一覧画面にプロジェクトの人気度が表示されていないことを言及した．検索結果の一覧画面では OSS の名前，機能の簡単な説明，分類，言語，ライセンスの 5 つの情報を表示しており，スターなどの人気度指標は表示していなかった．この情報は詳細画面で閲覧できるが，人気度を考慮する選択者のために，この情報を一覧画面に表示し，比較できるようにすることが必要である．

図 18 の結果が示すように，被験者はソフトウェア選択を行う上で Ellecto を最も高く評価した．とても良いと回答された割合は，OpenHub の約 3 倍，GAS & GitHub の約 6 倍で

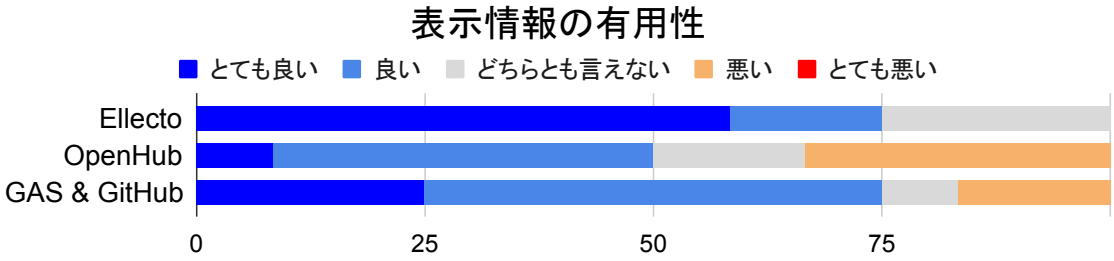

図 17: 検索結果の画面や詳細画面での表示情報の有用性のアンケート結果

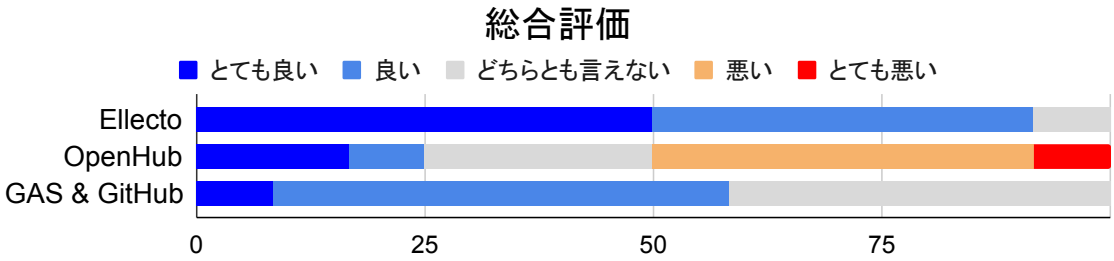

図 18: ソフトウェア選択を行う上での総合評価のアンケート結果

あった.

これらの結果から，被験者はソフトウェア選択を行う上で既存ツールよりも Ellecto を高く評価したことが分かった．得られた結果はキーアイデアや設計の妥当性を示し，Ellecto はソフトウェア選択を行う上で有益なツールであると考えられる.

5.3.2 全課題終了後のアンケート

被験者は，全ての課題終了後にもアンケートに回答した．ソフトウェア選択を行う上で使用したいと思う順に 3 つのツールを順位付けした結果を図 19 に示す.

また，自由記述形式で回答された Ellecto の良かった点と改善点には，主に以下のようなものがあった.

- Ellecto の良かった点
  - 検索オプションで更新の有無や開発期間の長さを直感的に指定できること
  - 検索オプションが簡潔かつ有用であること
  - 検索要素の重みを簡単かつ明快に変更できること
  - 必要な情報が的確に指定，表示されていたこと
  - 依存関係が分かりやすく，特定の依存関係を含むか調べられること

表 5: 各ツールごとの回答平均値

	Ellecto	OpenHub	GAS & GitHub
検索機能の使いやすさ	**4.17**	2.75	3.75
検索オプションの有用性	**4.42**	-	4.00
表示情報の網羅性	**4.50**	3.17	3.50
表示情報の有用性	**4.33**	3.25	3.83
総合評価	**4.42**	2.83	3.67

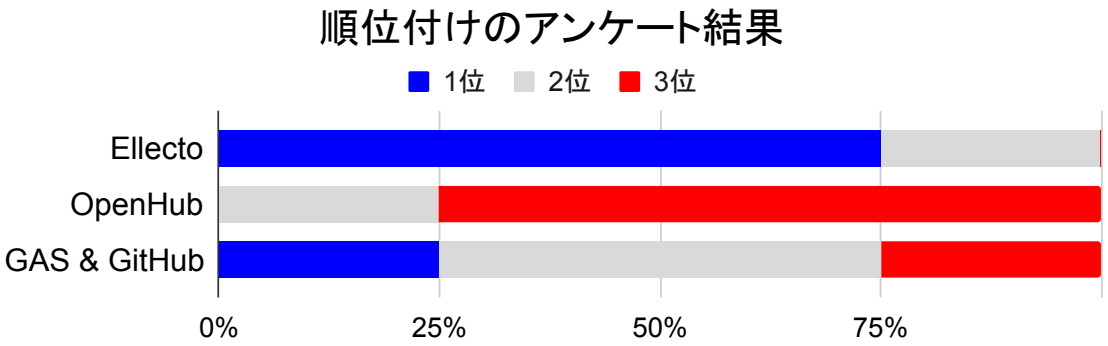

図 19: 順位付けのアンケート結果

  - 一画面に全ての情報がまとまっていたこと

- Ellecto の改善点
  - 言語やライセンスを選択式にして欲しいこと
  - 検索結果の表示画面にプロジェクトの人気度を表示していないこと
  - Section によるフィルタリングができるようにして欲しいこと
  - ソート機能で複数条件でのソートができるようにして欲しいこと

図 19 の結果から，被験者の 75%はソフトウェア選択時に Ellecto を使用したいと回答したことが分かる．良かった点として挙げられた，検索オプションの有用性や使いやすさ，必要な情報が表示されていたこと，依存関係が他のツールに比べて理解しやすかったことなどが結果に影響したと考えられる．これらは 3.1 節で述べたキーアイデアが有効であったことを示しており，Ellecto の設計の妥当性を表している．また，Ellecto を 3 位と回答した被験者は存在しなかった．このことから，Ellecto はソフトウェア選択を支援するツールとして一定の価値を示すツールであると考えられる．

言及された改善点は，Ellecto のさらなる機能性や利便性の向上の可能性を示した．言語とライセンスの指定はワード入力によって実現していたが，ライセンスなどの知識が乏しいまたはタイプミスを嫌う選択者にとって，選択式の方が好まれることが分かった．また，検索結果の表示画面にスターなどのプロジェクトの人気度を表示することで，より比較が容易になることも分かった．さらに，Section でのフィルタリングは機能での検索結果の絞り込みを実現できる．現状の機能では，ソフトウェアの機能は検索一覧画面や詳細画面で確認することが必要であるが，これらを Section で絞り込むことで，より効率的にソフトウェアを探せるのではないかと考える．最後に，複数条件でのソート機能は既存ツールでは実現されていない機能になるため，実装することで Ellecto の有効性を異なる角度から主張できると考える．この機能により，関連度が高く開発歴が長いソフトウェアや，関連度が高く最終更新日が近いソフトウェアなどを探すことが容易になり，ソフトウェア選択をより効率的に行えるようになる．

6　妥当性への脅威

6.1　外部妥当性

実施した被験者実験は，大阪大学に所属している学生 12 名を対象に実施した．そのため，普段から OSS を使用しているソフトウェア開発者などを対象に同様の実験をした場合，OSS の選択基準の違いなどの理由から異なる結果が得られる可能性がある．

また，実際のプロジェクトの要件 [3, 4, 13] を基に 3 つの課題を作成した．これらの選択ソフトウェアの種類や設定要件が結果に影響を与えた可能性がある．この脅威を軽減するために，3 つの課題で多様な要件のソフトウェアを選択することを促したが，より多くの要件での調査が必要である．

6.2　内部妥当性

今回の被験者実験では，被験者間のスキルレベルの違いがツールのアンケート結果に影響を与えた可能性がある．しかし，比較対象として用いた OpenHub と GAS，GitHub に対する被験者間の習熟度の差は少なく，この脅威を軽減するため，使用前にツールに慣れる時間を均等に取った．

また，ツールの使用順がアンケート結果に影響を与えた可能性も考えられる．この脅威を軽減するために，課題ごとの使用ツールや課題に取り組む順番を被験者ごとに設定した．しかし，被験者数は 12 名であるため，充分な数でグループ分けが出来たとは言い難く，実験者数を増やし，さらなる調査が必要である．

7　おわりに

本研究では，ソフトウェア選択者が実施すると考えられる作業を効率的に行うための，ソフトウェア選択支援ツールに必要な機能のキーアイデアを提案し，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 Ellecto を作成した．Ellecto では，Ubuntu と Debian が提供する OSS プロジェクトを対象にメタデータを収集し，システムを構築した．また，Ellecto について被験者実験を実施した．実験を通じて得られた結果から，Ellecto は既存ツールと比較して効率的にソフトウェアの選択を支援できることが分かった．検索機能の有用性や表示情報の網羅性，総合評価など，全てのアンケート項目において，とても良いと回答された割合が最も高かった．さらに，ソフトウェア選択時に使用したいツールを順位付けしたアンケート結果では，被験者の 75%が 1 位と回答し，最下位と回答する被験者はいなかった．

今後の課題は 2 つある．1 つ目はデータの拡張である．現在 Ellecto は Ubuntu と Debian の OSS プロジェクトの情報のみを検索・閲覧できる．そのため，他の 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，データを増やすことでより多くのソフトウェアの情報を提供できるようになる．また，Linux ディストリビューションに関わらず，その他のソフトウェアの情報を DB に登録できる機能を実装することも必要である．この機能により，多種多様なソフトウェアの検索・特徴の表示を行えるようになる．2 つ目は被験者実験で得られた改善点を Ellecto に反映することである．自由記述形式で回答された Ellecto の改善点を反映し，ソフトウェア選択をより効率的に支援することを目指したい．

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，本研究を行うにあたり，実験の妥当性に関する御助言など様々な御指導を賜りました．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 松下 誠 准教授には，研究の方向性の検討から研究活動の直接の御指導，論文の執筆に至るまで，研究活動全ての場面で手厚く御指導，御助言を賜りました．松下 誠 准教授の適切な御指導により，3 年間に渡り研究活動を行え，研究会での発表や論文の執筆をすることができました．心より深く感謝申し上げます．

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，研究室内での発表機会において，多くの御助言を賜りました．心より深く感謝申し上げます．

I would like to express my deepest gratitude to Professor Raula from the Department of Computer Science, Graduate School of Information Science and Technology, Osaka University, for his invaluable advice during presentations within our laboratory.

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メンバーには，被験者実験の参加や普段の日常生活において，様々な時間を共にしました．心より深く感謝申し上げます．

最後に，研究活動に関わらず，多くのサポートをしていただきました事務職員 軽部 瑞穂氏に心より深く感謝申し上げます．

参考文献

[1] Adewole Adewumi, Sanjay Misra, Nicholas Omoregbe, and Luis Fernandez-Sanz. FOSSES: Framework for Open - Source Software Evaluation and Selection. *Software: Practice and Experience*, Vol. 49, pp. 780 – 812, Feb 2019.

[2] Paris C. Avgeriou, Davide Taibi, Apostolos Ampatzoglou, Francesca Arcelli Fontana, Terese Besker, Alexander Chatzigeorgiou, Valentina Lenarduzzi, Antonio Martini, Athanasia Moschou, Ilaria Pigazzini, Nyyti Saarimaki, Darius Daniel Sas, Saulo Soares de Toledo, and Angeliki Agathi Tsintzira. An overview and comparison of technical debt measurement tools. *IEEE Software*, Vol. 38, No. 3, pp. 61–71, 2021.

[3] Bitnami. Discourse. `https://hub.docker.com/r/bitnami/discourse`.

[4] Bitnami. Redmine. `https://hub.docker.com/r/bitnami/redmine`.

[5] Elizabeth Bjarnason, Patrik Aberg, and Nauman bin Ali. 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.

[6] Ning Chen, Steven Hoi, Shaohua Li, and Xiaokui Xiao. SimApp: A Framework for Detecting Similar Mobile Applications by Online Kernel Learning. In *WSDM 2015 - Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, pp. 305–314, Jan 2015.

[7] S.R. Chidamber and C.F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, Vol. 20, No. 6, pp. 476–493, 1994.

[8] Glen Jeh and Jennifer 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.

[9] Shinji Kawaguchi, Pankaj K. Garg, Makoto Matsushita, and Katsuro Inoue. MUDABlue: An Automatic Categorization System for Open Source Repositories. *Journal of Systems and Software*, Vol. 79, No. 7, pp. 939–953, 2006.

[10] Valentina Lenarduzzi, Davide Taibi, Davide Tosi, Luigi Lavazza, and Sandro Morasca. Open Source Software Evaluation, Selection, and Adoption: A Systematic Literature

Review. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 437–444, 2020.

[11] Xiaozhou Li, Sergio Moreschini, Zheying Zhang, and Davide 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.

[12] Rensis Likert. A technique for the measurement of attitudes. *Archives of Psychology*, Vol. 140, pp. 1–55, 1932.

[13] mailcow. Postfix. `https://hub.docker.com/r/mailcow/postfix`.

[14] T.J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, Vol. SE-2, No. 4, pp. 308–320, 1976.

[15] Collin McMillan, Mark Grechanik, and Denys Poshyvanyk. Detecting Similar Software Applications. In *2012 34th International Conference on Software Engineering (ICSE)*, pp. 364–374, 2012.

[16] Kawser Wazed Nafi, Banani Roy, Chanchal K. Roy, and Kevin A. Schneider. A Universal Cross Language Software Similarity Detector for Open Source Software Categorization. *Journal of Systems and Software*, Vol. 162, p. 110491, 2020.

[17] Phuong T. Nguyen, Juri Di Rocco, Riccardo Rubei, and Davide Di Ruscio. CrossSim: Exploiting Mutual Relationships to Detect Similar OSS Projects. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 388–395, 2018.

[18] Phuong Nguyen, Juri Rocco, Riccardo Rubei, and Davide Di Ruscio. An automated approach to assess the similarity of GitHub repositories. *Software Quality Journal*, Vol. 28, pp. 595 – 631, 06 2020.

[19] Md Omar Faruk Rokon, Pei Yan, Risul Islam, and Michalis Faloutsos. Repo2Vec: A Comprehensive Embedding Approach for Determining Repository Similarity . In *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pp. 355–365, 2021.

[20] Bruno Rossi, Barbara Russo, and Giancarlo Succi. Adoption of free/libre open source software in public organizations: factors of impact. *Information Technology and People*, Vol. 25, pp. 156 – 187, Jun 2012.

[21] Riccardo Roveda, Francesca Arcelli Fontana, Ilaria Pigazzini, and Marco Zanoni. Towards an architectural debt index. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pp. 408–416, 2018.

[22] Mohamed Sarrab and Osama M. Hussain 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.

[23] Minaoar Hossain Tanzil, Gias Uddin, and Ann Barcomb. “How do people decide?”: A Model for Software Library Selection . In *2024 IEEE/ACM 17th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, pp. 1–12, Apr 2024.

[24] Ferdian Thung, David Lo, and Lingxiao Jiang. Detecting similar applications with collaborative tagging. In *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pp. 600–603, 2012.

[25] Ferdian Thung, David Lo, and Julia Lawall. Automated Library Recommendation. In *2013 20th Working Conference on Reverse Engineering (WCRE)*, pp. 182–191, 2013.

[26] Yun Zhang, David Lo, Pavneet Singh Kochhar, Xin Xia, Quanlai Li, and Jianling Sun. Detecting Similar Repositories on GitHub. In *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 13–23, 2017.

[27] 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.

付録 A　課題文

付録 A.1　課題 1

プロジェクト管理やタスク管理を行えるソフトウェア Redmine を使用したいと考えています．Redmine が使用できる環境の構築や必要な機能を揃えるために，以下の機能を満たすソフトウェアを選択してください．

1. データベースのクライアントツール

   データベースにアクセスしてクエリを実行したり，データベースの設定を管理したりするために使用できる，データベースのクライアントツールを選択してください．

2. YAML ファイルを操作するコマンドラインツール

   Redmine の設定ファイルで使用されている YAML ファイルを操作するためのコマンドラインツールを選択してください．選択するソフトウェアは以下の条件を満たす必要があります．

   - YAML を JSON に変換できること
   - Apache-2.0 ライセンスであること
   - 直近 1 年以内に更新があること

上記の条件を満たすソフトウェアを選択し，回答方法に従って回答シートに記入してください．

付録 A.2　課題 2

オンラインコミュニティを構築するためのプラットフォーム Discourse を使用したいと考えています．Discourse が使用できる環境の構築や必要な機能を揃えるために，以下の機能を満たすソフトウェアを選択してください．

1. データベースのクライアントツール

   データベースにアクセスしてクエリを実行したり，データベースの設定を管理したりするために使用できる，データベースのクライアントツールを選択してください．選択するソフトウェアは以下の条件を満たす必要があります．

   - PostgreSQL データベースと通信できるクライアント機能を持つこと
   - 開発履歴が少なくとも 10 年以上あること

- 直近半年以内に更新があること

2. データ圧縮用のライブラリ

   画像やスクリプトなどのデータを圧縮し，高速で効率的なデータ転送を可能にするソフトウェアを選択してください．選択するソフトウェアは以下の条件を満たす必要があります．

   - Python が使用できること
   - MIT ライセンスであること
   - 直近半年以内に更新があること

上記の条件を満たすソフトウェアを選択し，回答方法に従って回答シートに記入してください．

付録 A.3　課題 3

電子メールの送信が行えるメール転送エージェント Postfix を使用したいと考えています．Postfix が使用できる環境の構築や必要な機能を揃えるために，以下の機能を満たすソフトウェアを選択してください．

1. MySQL との連携

   MySQL を利用して仮想ドメインやメールボックスを動的に管理する機能を持った，Postfix の MySQL プラグインを選択してください．選択するソフトウェアは以下の条件を満たす必要があります．

   - 開発履歴が少なくとも 10 年以上あること
   - 直近半年以内に更新があること
   - ページ下部の依存関係リストに記載しているソフトウェアを依存関係として持たないこと

2. SMTP 認証をするためのモジュール

   メール送信時の SMTP 認証を行えるモジュールを選択してください．選択するソフトウェアは以下の条件を満たす必要があります．

   - SMTP AUTH を提供するために SASL 認証を利用できること
   - BSD-3-Clause ライセンスであること
   - 開発履歴が少なくとも 10 年以上あること

- 直近半年以内に更新があること
- ページ下部の依存関係リストに記載しているソフトウェアを依存関係として持たないこと

他に使用を決めているソフトウェアと依存関係の競合が起きないように，以下の依存関係リストに記載しているソフトウェアが実行時の依存関係に含まれないようにしてください．

依存関係リスト

- libssl3

  多くのネットワーク関連 OSS がこのソフトウェアの異なるバージョンに依存しているため，競合が発生しやすい
- redis-tools

  クライアント側のデータベースとして Redis を使用するため，他の Redis 関連ツールと競合が発生しないように
- syslog-ng-core

  多くのログ管理ツールがこのソフトウェアの異なる依存関係で競合を起こしているため，その心配をしないでいいように

上記の条件を満たすソフトウェアを選択し，回答方法に従って回答シートに記入してください．