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
SC2022-45(2023-03)

メソッド名構成単語に基づくソフトウェア概要推測に関する検討

寺川　航平† 　陳　　思楠† 　佐伯　幸郎†† 　中村　匡秀†,†††

† 神戸大学 〒 657–8501 神戸市灘区六甲台町 1–1
††† 理化学研究所・革新知能統合研究センター 〒 103–0027 東京都中央区日本橋 1–4–1
†† 高知工科大学 〒 782–8502 高知県香美市土佐山田町宮ノ口 185
E-mail: †odajin@ws.cs.kobe-u.ac.jp, ††chensinan@gold.kobe-u.ac.jp, †††saiki.sachio@kochi-tech.ac.jp,
††††masa-n@cmds.kobe-u.ac.jp

あらまし　ソフトウェア開発の現場では，過去に開発されたプロダクトが技術的負債を抱えていたり，資産的に管理されていないために再利用が困難である状況がしばしば存在する．そこで，我々は既存プロジェクトが抱える知的資産を新規開発に活かすソフトウェアアップサイクルについて検討を進めている．アップサイクルを促進するためには，説明書のないソフトウェアの概要を把握し，参照しやすくすることが課題となる．先行研究として，$corpus_{Pj}$ を用いたシステムの概要推測手法を提案した．クラス名構成単語を収集し，システムの概要推察にどの程度役立つかについて検証を行った．本研究では，さらなる精度向上を目指しメソッド名に着目する．$corpus_{Pj}$ 中の単語を重要度にもとづき重み付けし，Tag cloud 画像を用いて可視化することで効率的な概要把握を試みる．研究室内部で管理する 136 個の Java プロジェクトに対し提案手法を適用する．ソフトウェアが提供する働き，ソフトウェアを動作させる技術の 2 つの観点から効果を考察する．

キーワード　ソフトウェア開発, 技術的負債, アップサイクル, コーパス

Inferring Project Description Based on Method Name Elements

Kohei TERAKAWA†, Sinan CHEN†, Sachio SAIKI††, and Masahide NAKAMURA†,†††

† Kobe University, Rokkodai-cho 1–1, Nada-ku, Kobe, Hyogo, 657–8501 Japan
††† Riken AIP, 1–4–1 Nihon-bashi, Chuo-ku, Tokyo, 103–0027 Japan
†† Kochi University of Technology, 185 Tosayamadacho Miyanokuchi, Kami, Kochi, 782–8502 Japan
E-mail: †odajin@ws.cs.kobe-u.ac.jp, ††chensinan@gold.kobe-u.ac.jp, †††saiki.sachio@kochi-tech.ac.jp,
††††masa-n@cmds.kobe-u.ac.jp

Abstract　In the domain of software development, there are often circumstances in which past products prove difficult to repurpose due to technical debt or poorly managed assets. As a result, we are currently exploring the concept of software upcycling, which involves leveraging the intellectual assets of existing projects for fresh development. To facilitate the upcycling process, it is imperative to grasp an overall understanding of software without any documentation and to make it easily referenceable. In a previous study, we proposed a methodology for deducing the system overview by means of $corpus_{Pj}$. We extracted component words from class names and validated the utility of $corpus_{Pj}$ in terms of deriving the system overview. In this study, we concentrate on the method name elements in order to further enhance accuracy. Words in $corpus_{Pj}$ are given importance-based weighting, and represented through Tag cloud images to achieve efficient overview. We implement the suggested methodology across 136 Java projects managed within our laboratory, and scrutinize its efficacy from two angles: the functions delivered by the software, and the underlying technologies that enable the software to function.

Key words　Software Development, Technical debt, Upcycling, Corpus

1. はじめに

近年，デジタル技術の発展により様々な場所でソフトウェアプロダクトが活用されている．ソフトウェアを取り巻く環境の変化はめざましく，技術進歩やビジネス要求の変化といった事情から，ソフトウェアには次第に技術的負債 [1] が蓄積されていく．そうした現状への解決策として，我々はソフトウェアアップサイクルについて検討を進めている．ソフトウェアアップサイクルとは，既存プロジェクトに隠された知的資産を新規のソフトェア資産として転換することで，開発効率の向上と既存知識の活用を目指す考え方である．

一方で，ソフトウェア開発の現場には資産として保守・管理されていないリポジトリが存在する．設計や実装のアイデアに再利用の価値があるにも関わらず，*README* 等のプロダクトにまつわる説明書が欠けていたり，不十分な場合は再利用の際にコストが生じる．そのため，アップサイクルを行う上では，説明書のないソフトウェアを資産化し，参照しやすくすることが課題となる．

我々は先行研究として，プロジェクトコーパス ($corpus_{Pj}$) を用いたシステムの概要推測手法を提案した [2]．クラス名はシステムの目的や機能に関連すると仮定し，クラス名構成単語を収集することでソフトウェアの文脈におけるコーパスを作成した．これを用いて被験者実験を行い，$corpus_{Pj}$ がシステムの概要を反映していることを確かめた．

しかしながら，先行研究にはいくつか課題が残されている．１つのクラス内に多くのメソッドを抱えるゴッドクラスや，クラス名が動作に関わる情報を含んでいない場合，推測に十分な情報が得られないことが判明した．そのため，クラスが内部に抱える情報も $corpus_{Pj}$ に含める必要がある．

本研究では，先行研究より把握精度を向上させるために新たな概要把握手法を提案する．キーアイデアとしてメソッド名構成単語に着目し，これらの単語がシステムの働きを反映しているという仮説を立てる．メソッドはいずれのシステムにも必ず備わっている情報であり，メソッド名のみから効果的に意味を抽出できればソフトウェアプロダクトの概要把握に費やす時間を削減し，ソフトウェアアップサイクルの促進が見込める．

提案手法は以下の 4 つのアプローチから構成される．

（A1）リポジトリマイニング

（A2）メソッド名構成単語の取得

（A3）tf-idf による重み付け

（A4）Tag cloud 画像による重要度の可視化

研究室で管理する 136 個の Java プロジェクトを対象に提案手法を用いて実験的評価を行う．プロジェクトの概要を構成する要素として以下の２つの観点から考察を行う

What: ソフトウェアが提供する働きや提供する価値

How: ソフトウェアを動作させるための技術

結果として，計 14604 個のメソッドと計 1659 個のメソッド名構成単語を得た．What と How の観点から考察した結果，メソッドのもつ「名詞+動詞」の構造のうち，名詞として登場する要素がシステム中で扱われるモノ・コトに該当し，これらの情報が What の観点から概要推測に有用であると判明した．一方で，デザインパターンといった高レイヤの情報をメソッド名から取得することは困難であるとわかった．

2. 先行研究:プロジェクトコーパスを用いたソフトウェア概要推測手法の実験的評価 [2]

2.1 着目する課題

組織や会社などのソフトウェア開発現場において，全てのプロダクトが資産的に管理されているわけではない．*README* といったプロダクトにまつわる説明書が欠けている，または不十分であるといったケースは多く存在する．一方で，そうしたソフトウェアには資産的価値が含まれている場合がある．具体的には，設計や実装のアイデアの全体，もしくはその一部である．こうした既存ソフトウェアの価値に着目し，新たに手を加えることで価値あるソフトウェア資産に転換することをソフトウェアアップサイクル [3] と呼ぶ．アップサイクルを促進させるには，説明書の有無に関わらずプロジェクトの概要把握を行うことが課題となる．

2.2 先行研究における提案手法

先行研究では，既存プロジェクトを *README* などの説明書に頼らず概要把握を行う手法を提案している．クラス名構成単語から $corpus_{Pj}$ を作成し，これによりシステムのもつ機能・使用技術・概要を推測を行う．

先行研究で行った実験は，大きく分けて 5 つの Step で構成される．まず，Step1 でプロジェクト・被験者の選定を行う．研究室内で管理する Gitlab に含まれる Java プロジェクトを採用した．Step2 ではコーパスの準備を行う．リポジトリマイニングによりプロジェクトに含まれるクラス名を取得し，単語単位で文字列を分割する．これによって得られた単語群を先行研究では $corpus_{Pj}$ と定義した．Step3 では質問の設計を行う．プロジェクトへの理解度を図るため，以下の 3 つの質問を設計した．

（Q1）：このシステムにはどんな機能があるか，思いつくだけ箇条書きで書いてください

（Q2）：このシステムはどのような技術や仕組みで動作しているか，思いつくだけ箇条書きで書いてください

（Q3）：$corpus_{Pj}$ から推察される，システムの概要を 1〜2 行程度で記述してください．

Step4 では被験者による回答を行う．得られた単語群を辞書順に表示することに加え，出現頻度による重み付けを用いた Tag cloud 画像を提示する．Step5 では回答の採点を行う．各プロジェクトの詳細を把握している Gitlab の管理者が回答に対して採点を行う．

2.3 プロジェクトコーパス

章 2.1 の先行研究における提案手法では，クラス名分割処理によって得られた単語群を $corpus_{Pj}$ と定義した．コーパス [4] とは一般に言語学の分野で用いられ，テキストや発話を大規模に集めてデータベース化した言語資料を指す．

つまり，$corpus_{Pj}$ とはソフトウェアの文脈におけるコーパスを意味する．具体的には，プロジェクトに含まれるクラス名を収集し単語に分割する．得られた単語群は抽出元のプロジェ

クトの性質を反映しているとみなし，これを参考にプロジェクトの概要を推測できれば，*README* が存在しないプロジェクトについても概要把握が容易に行える．

2.4 先行研究における実験の結果

実験では 12 名の被験者から回答を得た．その結果，プロジェクトの推測は 2 つの要因に左右されることが確認できた．1 つ目は，$corpus_{Pj}$ がシステムの持つ情報をどれだけ正しく反映しているかという点である．１つのクラスの中にどれだけの機能や役割をもたせるかは開発者によって異なる．クラス内部のメソッド名を取得することで，クラスの担う機能や役割をより正確に把握できることが期待される．

2 つ目は，単語のもつ情報量である．多く出現する単語が必ずしもソフトウェアにとって重要ではない例を確認した．そのため，ソフトウェアの特徴を表す単語に重みをつけて表示させることで，より効果的な概要把握が可能になると考えられる．

3. 提 案 手 法

3.1 目的とキーアイデア

本研究での目的は，先行研究とは異なるアプローチを取り，把握精度を向上させることである．キーアイデアは 2 つある．1 つ目は，メソッド名を用いた $corpus_{Pj}$ の作成である．メソッド名はソフトウェアの機能に深く関連するため，先行研究におけるアプローチと比較してより詳細にシステムの動作が把握可能となることが期待される．2 つ目は，$corpus_{Pj}$ を単語の重要度によって重み付けすることである．先行研究では単語の出現頻度を重み付けの指標として採用したが，必ずしも多く登場する単語がシステムの概要を表すとは限らないことを確認した．そのため，本研究では別の尺度を用いて単語の重要度を決定する．

3.2 アプローチ

提案手法では，4 段階に分けたアプローチを行う．概要を図 1 で表す．アプローチの詳細は下記の通り，(A1) (A4) で構成される．

(A1) リポジトリマイニング: 対象となるプロジェクト（Pj）のリポジトリについて，リポジトリマイニングによって存在するメソッド名を全て抽出する

(A2) メソッド名構成単語の取得: 構文解析ツールを用いて，ファイル内に含まれるメソッド名を取得する．その後，得られたメソッド名を単語に分解する．

(A3) tf-idf による重み付け: 得られた単語群に対し，tf-idf を用いて重み付けを行う．これにより，ソフトウェアの概要を表す特徴的な単語を明らかにする．

(A4) Tag cloud 画像による重要度の可視化: 各プロジェクトについて，単語の重みを Tag cloud を用いて可視化する．重要度の高い単語は大きく，強調して表示させる．

3.3 （A1） リポジトリマイニング

対象のプロジェクト群から，クラスファイルを取得する．ここでは環境設定ファイルや *README* といったファイルは除外し，処理の記述されたクラスファイルのみを対象としてマイニングを行う．

以下に，具体的なサービスを用いて例示する．我々の研究室で運用中の，MP3 形式の音源を再生するサービスである **MP3PlayService** に対してリポジトリマイニングを行うと，表 1 のような結果となる．なお，以下のステップにおいても同様に MP3PlayService を例に用いるものとする．MP3PlayService は Java 言語で書かれたプロジェクトであるため，拡張子が.java であるファイルのみを取得する処理を行う．

表 1 MP3PlayService に含まれる Java ファイル

クラス名
MP3List.java
MP3ListPlayer.java
MP3Music.java
MP3PlayService.java
MP3Player.java
Main.java

3.4 （A2） メソッド名構成単語の取得

構文解析を行うツールを用いて，ファイル内に含まれるメソッド名を取得する．その後，対象となるプロジェクトで用いられるプログラミング言語に対応した解析機を使用することで，メソッド名を文字列として取得する．得られた文字列に対し，そのプログラミング言語で用いられる命名規則に従い，正規表現等を用いて単語に分割する処理を施す．

表 1 中の MP3ListPlayer.java に着目する．表 2 は，ANTLR [5] を用いて MP3ListPlayer.java に含まれるメソッド名を取得した結果である．ANTLR とは，構文解析器を生成するためのパーサジェネレータであり，様々なプログラミング言語がサポートされている．

表 3 に，表 2 で得られたメソッドを分割した結果を示す．Java におけるメソッド名の命名規則は camelCase であるため，大文字出現位置を区切りとして分割する処理を行う．つまり，MP3ListPlayer.java からは，get, Instance, set, URL, List, URL, Music, play, stop, の 8 つがコーパスとして生成されることとなる．

表 2 MP3ListPlayer.java に含まれるメソッド名

メソッド名
getInstance
setURL
getListURL
getMusicURL
playList
stop
next

3.5 （A3） tf-idf による重み付け

A2 で得られた各単語について，tf-idf [6] を用いて重みを決定する．ソフトウェアプロジェクトを文書とみなし，プロジェクト単位で見たときの単語の頻度を tf(term frequency), プロジェクト単位で見たときの単語の頻度を idf(inverse document frequency) として計算する．以下に定義式を示す．

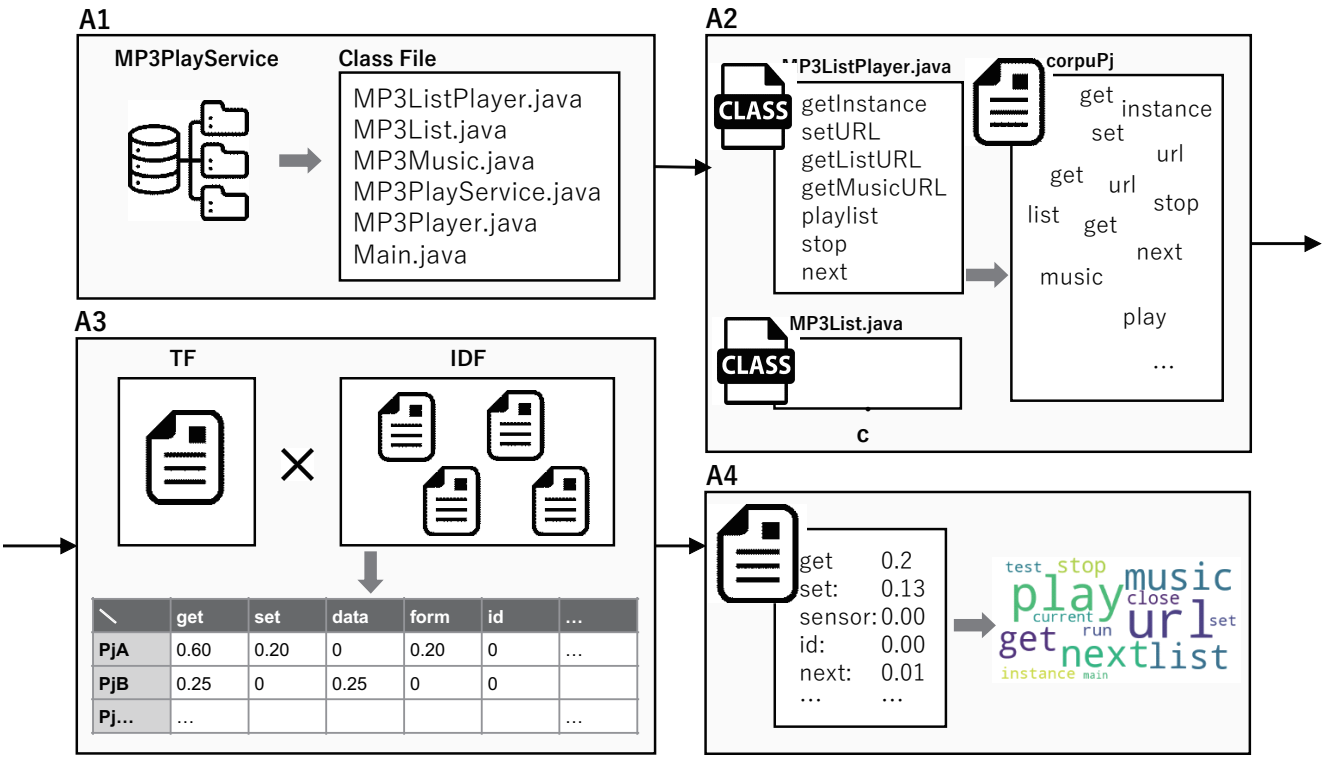


図 1　提案アプローチの流れ

表 3　メソッド名分割の結果

メソッド名	コーパス		
getInstance	get	Instance	
setURL	set	URL	
getListURL	get	List	URL
getMusicURL	get	Music	URL
playList	play	List	
stop	stop		
next	next		

$$tf(w,d) = \text{プロジェクト } p \text{ 内での単語 } w \text{ の出現回数}$$

$$idf(w) = \log \frac{\text{すべてのプロジェクト数}}{\text{単語 } w \text{ が出現するプロジェクト数}}$$

3.6 （A4） Tag cloud 画像による重要度の可視化

単語ごとに算出された重要度を，Tag cloud [7] 画像によって可視化する．重要度の大きな単語ほど大きく表示されるため，システムにとって重要な単語の判別が容易になることが期待される．

4. 評価実験

4.1 実験目的

本実験では，所属研究室の Gitlab で管理する 136 個の Java プロジェクトに対して提案手法を適用する．これは，Gitlab に存在する計 476 個のプロジェクトのうち，Java プロジェクトが最も多く存在するためである．また，計 42 人の開発者がこれらのプロジェクトの開発に関与している．得られた $corpus_{Pj}$ をもとに考察を行い，提案手法の有効性を確かめる．

4.2 手　　順

提案手法に従い，クラス名構成単語から $corpus_{Pj}$ を作成する．A1 のリポジトリマイニングでは，python を用いて GitlabAPI を利用した．A2 のメソッド名構成単語の取得では，解析ツールとして ANTLR を使用した．分割処理については，Java で用いられる camelCase 記法を踏まえて大文字出現位置を基準に正規表現を用いて分割処理を行った．A3 の tf-idf による重み付けでは，tf-idf 値の計算に python ライブラリの scikit-learn [8] を利用した．A4 の Tag cloud 画像の生成には，python ライブラリの wordcloud [9] を利用した．

4.3 結　　果

実験の結果，計 14604 個のメソッド名と計 1659 種類のメソッド名構成単語を取得した．単語ごとに出現回数のばらつきがあり，メソッド名によく用いられる単語やそうでない単語が存在することがわかった．

図 2 に，出現頻度の高いメソッド名構成単語を上位 20 位まで示す．最も多く出現した単語として，get が挙げられる．次いで set や create, update といった動詞が見られた．また，接続詞や副詞も多く見られた．特に，and, is, not, than といった単語が頻出しており，命名の際によく用いられていることがわかる．

一方で，名詞は総じて出現頻度が低いことがわかった．$corpus_{Pj}$ 中の大部分が固有名詞で構成されており，各プロジェクト中でシステムが取り扱うオブジェクトがメソッド名の中に名詞として現れていることがわかった．

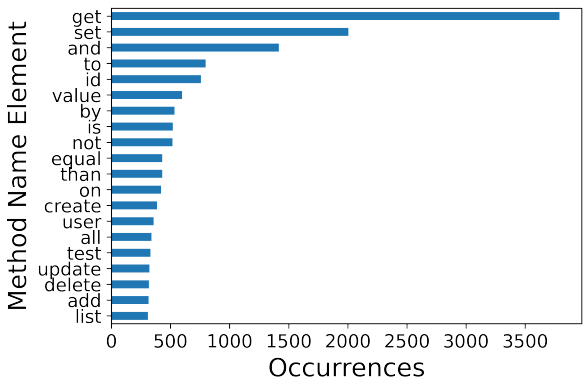


図 2　出現頻度の上位 20 位

5. 考　　察

5.1　考察観点の設定

プロジェクトは様々な要素から構成されており，概要を説明する際にはいくつかの視点が存在する．本研究では，プロジェクトの概要は以下の 2 点から説明できるものとする．

What: システムが提供する働き．ソフトウェアが可能にする動作であり，提供する価値にあたる．ソフトウェアの使用方法は使い手によって異なるため，そのシステムのユースケースとは区別する．プロジェクトに複数存在しうる．

How: システムを動作させる技術．システム構築のための手段にあたる部分であり，ライブラリやデザインパターンなどがこれにあたる．

この 2 つの観点から提案手法の効果について考察する．MP3PlayService を例に考える．このサービスにおける What とは，MP3 音源を再生できること，リスト再生ができることなどである．How については，MP3 ファイルを再生するために用いられる JavaLayer, HTML 形式をエンコードするために用いられる URLEncoder などがこれにあたる．表 4 に，考察で用いるプロジェクトの概要と $corpus_{Pj}$, Tag cloud 画像を示す．

5.2　What の観点

tf-idf により，システムが扱うオブジェクトを表す単語に重み付けすることができた．例として LifeActivityMailer を挙げる．このプロジェクトではセンサーによって各種のスコアを計測し，メールで送信する働きを持つ．画像を見ると score, sensor といった単語に重みがついており，システムの働きに関与するオブジェクトが強調して表示されている．

また，クラス名を利用する先行研究での手法と比較して，よりシステムに関する詳細な情報が取得できた．BLEAdapter に着目すると，先行研究の Tag cloud 画像では単語数が少なく，どういったデータを扱うかまでは推測できない．しかし，提案手法の Tag cloud 画像では location, message, date といった具体的にセンサーで扱う情報を読み取ることができる．このため，クラス内部に多くメソッドを抱えるシステムにおいて今回の提案手法が有効であることがわかった．

一方で，重要でない単語に重みがついた例も存在する．例として ENISHI に着目すると，get, set といった単語の重要度が高いことがわかる．これはシステムが SpringBoot を用いて構築された web アプリであることに関係しており，MVC モデルにおける Controller 層のクラス内で，データ取得のルーティングを行う記述が多いことが要因として挙げられる．また，entity として各クラス内に getter, setter メソッドが用意されていることなどからも，web アプリの実装パターン上頻出する動詞が存在する事がわかる．

以上を踏まえると，メソッドのもつ「名詞+動詞」の構造のうち，名詞として登場する要素がシステム中で扱われるモノ・コトに該当し，これらの情報が概要推測に有用であると考えられる．

5.3　How の観点

システムの使用技術やフレームワークが推測可能であるかについて考察する．例として，NewLINEAgentService に着目する．このプロジェクトは LINE Bot [10] を使用して構築されており，LINEBOTController.java クラス内で LINE Bot とのデータ授受や AccessToken のやり取りを行っているため，Tag cloud 画像の中にそれらの単語を確認することができる．しかしながら，重要度の値としては小さく，比較的小さく Tag cloud 画像上に表示されてしまっている．このように，単一の重要度の指標を用いた Tag cloud による可視化では，複数の機能概要を効率的に取得することは困難である．

また，デザインパターンといったシステムの設計に関する情報を推測することは困難である．thin-cas では実装において DAO パターンや Facade パターン [11] が用いられている．これらは主にディレクトリ名やクラス名での命名に現れているが，メソッド名からはこうした情報は読み取れない．このように，設計思想といった高レイヤの情報をメソッド名から取得することは困難である．

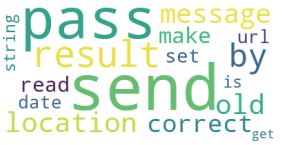
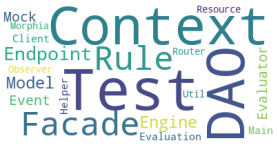
6. ま　と　め

本研究では，先行研究におけるプロジェクト概要把握の精度を向上させることを目的として，メソッド名構成単語に着目した手法を提案した．そして，$corpus_{Pj}$ に対して tf-idf を用いた重み付けを行い，単語の重要度を計算した．結果として，先行研究と比較してよりプロジェクトの内部動作にかかわるモノ・コトを知ることができた．それに伴い，プロジェクトが提供する働きについて概要把握を行う際に活用できる可能性を見出した．しかしながら，先行研究と比較して劣る点も存在する．デザインパターンなど高レイヤの設計に関わる情報を得ることが困難であった．

課題として，使用技術の観点では先行研究に劣ることが明らかとなった．そのため，今後は複数のアプローチから $corpus_{Pj}$ を作成し，目的に応じた概要推測の手法を検討したい．

謝辞　本研究の一部は JSPS 科研費 JP19H01138, JP20H05706, JP20H04014, JP20K11059, JP22H03699, JP19K02973, 特別研究員奨励費 22J13217，および，立石科学技術振興財団の研究助成を受けて行われている．

表 4　実験データの一部

概要	先行研究	提案手法
NewLINEAgentService LINE Botを使って，LINEメッセージを送信するサービス		
LifeActivityMailer 環境センサに変化があった際にメールで通知するサービス		
ENISHI Enishiのバックエンドサービス モック用に機能がかなり絞られている		
MP3PlayService 与えられたURLに基づき，MP3を再生するサービス 再生リストによる再生も可能		
BLEAdapter BLE（Bluetooth Low Energy）のデータを整形し，すれ違いフレームワークに適合させるアダプタ		
thin-cas ECAルールによるコンテキストアウェアサービス(CAS)を実行する軽量なコマンドラインアプリケーション		

文　　献

[1] P. Kruchten, R.L. Nord, and I. Ozkaya, “Technical debt: From metaphor to theory and practice,” Ieee software, vol.29, no.6, pp.18–21, 2012.

[2] 寺川航平，陳　思楠，中村匡秀，“プロジェクトコーパスを用いたソフトウェア概要 推測手法の実験的評価，” 電子情報通信学会技術研究報告，第 121 巻，pp.90–96，March 2022．オンライン．

[3] B. Bridgens, M. Powell, G. Farmer, C. Walsh, E. Reed, M. Royapoor, P. Gosling, J. Hall, and O. Heidrich, “Creative upcycling: Reconnecting people, materials and place through making,” Journal of Cleaner Production, vol.189, pp.145–154, 2018.

[4] T. McEnery, Corpus linguistics, Edinburgh University Press, 2019.

[5] “Antlr,” `https://www.antlr.org/`. (Accessed on 2/21/2023).

[6] L.-P. Jing, H.-K. Huang, and H.-B. Shi, “Improved feature selection approach tfidf in text mining,” Proceedings. International Conference on Machine Learning and Cybernetics, vol.2, pp.944–946vol.2, 2002.

[7] S. Bateman, C. Gutwin, and M. Nacenta, “Seeing things in the clouds: the effect of visual features on tag cloud selections,” Proceedings of the nineteenth ACM conference on Hypertext and hypermedia, pp.193–202, 2008.

[8] “scikit-learn machine learning in python,” `https://scikit-learn.org/stable/`. (Accessed on 2/20/2023).

[9] “Wordcloud for python documentation,” `http://amueller.github.io/word_cloud/`. (Accessed on 2/20/2023).

[10] “Line developers messaging api,” `https://developers.line.biz/en/docs/messaging-api/`. (Accessed on 2/21/2023).

[11] W. Pree, Design patterns for object-oriented software development, ACM Press/Addison-Wesley Publishing Co., 1995.