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
KBSE2023-45,SC2023-28(2023-11)

LLMを活用したソフトウェアアップサイクルにおける素材抽出手法の検討

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には，組織特有の知見や組織内でのみ運用されるソフトウェアが存在する．そうした過去の開発プロジェクトに蓄積された知識をアップサイクルすることで，開発効率の向上と既存知識の活用を目指すソフトウアアップサイクルという考え方がある．しかし，ソフトウェア開発の背景や技術知識を持たない人がアップサイクル素材を抽出することは困難である．そこで本研究では，LLM を活用して個人の能力・知識に依存しないアップサイクル素材の抽出手法を提案する．評価実験として，素材抽出手法の正確性・網羅性・アップサイクルの文脈に対する適切性を検証する．具体的には，3 つのソフトウェアアップサイクルにおけるユースケースを検討する．それぞれについて提案手法を用いた素材抽出を行い，結果の考察を行う．

キーワード　ソフトウェア開発, 技術的負債, ソフトウェアアップサイクル, LLM

Study of Extracting Material in Software Upcycling with LLM

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 field of software development, there exists domain-specific knowledge and software that is used only within specific organizations. The concept of software upcycling aims to improve development efficiency and leverage existing knowledge by repurposing knowledge accumulated from such knowledge and software. However, extracting upcycling materials from such softwares is difficult for individuals who lack a background in software development and technical knowledge.Therefore, this study proposes a method for extracting upcycling materials that is not dependent on individual abilities and knowledge by utilizing Large Language Models (LLMs). As part of the evaluation experiment, the accuracy, comprehensiveness, and appropriateness of the upcycling material extraction method within the context of upcycling are examined. Specifically, we considered three software upcycling use cases. We applied the method to extract materials for each case.

Key words　Software Development, Technical debt, Software Upcycle, LLM

1. はじめに

近年の技術進歩に伴い，様々な組織・会社で新たなソフトウェアの開発が進められている．ソフトウェアを取り巻く環境変化に対応する形で，開発現場は新規開発や機能追加といった対応を迫られる．同時に，ソフトウェアの開発持続性を考慮しソフトウェアの保守や管理，ドキュメント化といった作業に適切な人的資源を充てる必要がある．こうした保守・運用にリソースが割かれない現場では，次第に技術的負債が蓄積してゆく [1].

こうしたソフトウェアの開発運用の課題を解決する考え方として，**ソフトウェアアップサイクル**がある [2]．ソフトウェアアップサイクルは，既存プロジェクトに隠された知的資産を新

規のソフトウ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．ソフトウェアアップサイクルは既存プロジェクトからのアップサイクル素材の抽出と抽出した素材を組み合わせる 2 つのステップが存在する [3]．特に，素材抽出のステップではプログラミング言語やライブラリといった技術に関する知識や，組織内特有の暗黙知やライブラリへの理解などが求められる．

しかしながら，組織内でのソフトウェア開発背景や技術知識を持たない人がアップサイクル素材を抽出することは困難である．ソフトウェアが作られた経緯や意図，実装の詳細，管理方法といった情報が明らかでない場合や，用いられる技術に対する知識を備えていなければ，ソースコードを理解するために多くのコストがかかる．ソフトウェアアップサイクルを促進するためには，アップサイクルを行う人の知識や能力に依存することなく，既存ソフトウェアから適切にアップサイクル素材の抽出を行うことが課題である．

本研究では，**非公開情報やドメイン知識を伴う組織内ソフトウェアからアップサイクル素材を抽出する**こと目的とする．キーアイデアとして，近年普及が進んでいる**大規模言語モデル（LLM）**の一種である GPT モデルを利用する．具体的には，以下の 2 つのリサーチクエスチョンを設定する．

RQ1　アップサイクル素材抽出プログラムを用いて，組織内ソフトウェアにまつわる正確な情報取得は可能か？

RQ2　アップサイクル素材抽出プログラムを用いて，組織内ソフトウェアからニーズに基づいた網羅的な素材抽出は可能か？

評価実験として，神戸大学中村研究室内で管理・運用しているソフトウェアに対し提案手法を適用する．ソフトウェアアップサイクルに関する 3 つのユースケースを想定して，アップサイクル素材の正確性や網羅性，ユースケースの文脈に対する適切性などを評価する．

2. 準　　備

2.1 技術的負債

技術的負債とは，ソフトウェア開発運用を行う中で蓄積する改修しなければならないプロダクトの全体，あるいはその一部である．不具合（バグ）だけでなく，十分な作業時間が確保できなかった等の理由で，作成されたものの改善の余地が多分に残るプロダクトも技術的負債に含まれる．

ソフトウェア開発を行う組織内には，特有のドメイン情報や非公開のソフトウェア等が存在する．本稿では外部への公開を想定しない，特定の組織内でのみ開発・運用を行うソフトウェアを組織内ソフトウェアと呼ぶ．組織内ソフトウェアは組織特有のコンテキストや非公開情報を含む場合が多く，仕様の文書化や説明の不十分さが属人化を進行させる要因となる．こうした組織内ソフトウェアに関する仕様や実装の理解を得るためには，多大なコストを掛けて解読する必要がある．

2.2 ソフトウェアアップサイクル [2]

ソフトウェアアップサイクルとは，既存プロジェクトの一部分であるコード・設計・文書といった素材を，アップサイクルによって新規の価値あるソフトウェア資産に転換することである．特に，アップサイクルを図る素材をソフトウェアアップサイクル素材と呼ぶ．具体的にはソフトウェアにまつわる一般的な知識・情報や特定の組織内でのみ運用される知識・情報を指す．

ソフトウェアアップサイクルは，**アップサイクル素材の抽出とアップサイクル素材の組み合わせ**の 2 つの工程に分かれている．ソフトウェアアップサイクルにおける課題は，素材抽出の成否が個人の能力・知識に依存することである．プログラムコードを読み解き，適切な素材を選び取るには様々な知見が必要となる．具体的には，プログラミング言語やライブラリといった技術に関する知識や，組織内特有の暗黙知やライブラリへの理解などが挙げられる．

2.3 予備実験大規模言語モデル (LLM)

大規模言語モデル (LLM) [4] は，大量のテキストデータでトレーニングされた自然言語処理モデルである．自然言語を用いて様々な処理を行う事が可能であり，チャットボットや検索エンジン，翻訳といった幅広い領域で利用されている．ソフトウェア開発の分野でも利用が広まっており，OpenAI 社と GitHub 社が共同で開発した Github Copilot というツールでは，コード補完や解説といった機能が提供されている [5]．

一方で，LLM は非公開情報やドメイン知識を伴う出力の生成に対してハルネーションを起こす傾向がある [6]．このような LLM の学習データに存在しない知識に関する生成を行う手段として，In-context Learning がある．In-context Learning では，プロンプト内に出力生成の際に参照させる情報を併せて入力する．多くの LLM には入力できるプロンプト長に上限が存在するため，トークン上限の範囲内で事前知識をプロンプトに含める必要がある．

3. 提 案 手 法

3.1 目的とキーアイデア

本研究の目的は，プログラムの実行を通じて，組織内ソフトウェアのソースファイルからソフトウェアアップサイクル素材を抽出することである．キーアイデアは，LLM を活用したソフトウェアアップサイクル素材の抽出である．本研究のアプローチは次のとおりである．

（A1）　アップサイクル素材の定義
（A2）　アップサイクル素材の抽出を行うプロンプトの設計
（A3）　LLM を用いた素材抽出アルゴリズムの構築

3.2 (A1) アップサイクル素材の定義

ソフトウェアアップサイクル素材とは，ソフトウェアに含まれるコード・設計や実装のアイデアといった情報を指す．しかし，すべての人にとって価値を持つアップサイクル素材は存在しない．素材の有用性は，アップサイクルを行う人やコンテキストに左右される．例えば，ある技術に関する周辺知識を持たない人にとっては，その技術を用いて作成されたソフトウェアは技術の使用例として参照されうるため，再利用価値を持つことになる．

3.3 (A2) アップサイクル素材抽出のためのプロンプト設計

LLM を用いてアップサイクル素材抽出を行うためのプロン

プトを設定する．プロンプトは一般的に以下の要素で構成される [7] [8].

- 命令: LLM に実行してほしい特定のタスクまたは命令
- 文脈: 前提条件や背景情報を含む文章
- 入力データ: 出力を得るために必要な入力や質問
- 出力指示子: 出力のタイプや形式

アップサイクル素材を抽出する状況に応じた命令・文脈・入力データの設定を行う．3.2 で述べたように，アップサイクル素材はアップサイクルを行う状況やコンテキストに依存する．アップサイクルを行う人が持つニーズや知識量に応じたプロンプトを作成できるように，プロンプトの雛形を図 1 のように定める．

文脈の設定として，context 部分でソフトウェアアップサイクルの経緯を表す．当事者の持つニーズ (upcycle_needs) とそれに関連する情報 (related_information) を当事者の把握する範囲で記述する．命令の設定として，instruction 部分で定型文の指示を行う．文脈に基づき，関連する情報の抽出を指示する．文脈としては，アップサイクルを行う人が知りうる情報や経緯を記述する．入力データの設定は input 部分で行う．ここには，アップサイクル素材の抽出対象となるソースファイルが埋め込まれる．出力指示子として，抽出した情報をリスト形式で出力するよう指示を与える．

3.4 (A3)LLM を用いた素材抽出アルゴリズムの構築

素材抽出アルゴリズムは以下の要素で構成される．処理の流れを図 2 に示す．

Step 1. リポジトリマイニング
Step 2. プロンプトの雛形の定義
Step 3. プロンプトの作成
Step 4. ソースファイルごとの LLM 実行

Step1 では，対象となるソフトウェアのリポジトリに対してリポジトリマイニングを行う．Step2 では，LLM へ入力するプロンプトの雛形を作成する．3.3 で検討した項目に基づき，アップサイクルの文脈に応じた内容を作成する．Step3 では，Step2 で作成したプロンプトの雛形にソースファイルを埋め込む．Step4 では，Step1 で取得したソースファイルに対応するすべてのプロンプトを LLM に入力する．

4. 実　　装

リポジトリマイニングの対象は，神戸大学中村研究室で運用されている Gitlab である．Pyhon から Gitlab API を用いてソースファイルを取得した．Step2, 3 については Python と LangChain というライブラリを用いて実装した．LangChain [9] とは，LLM を利用して効率的にアプリケーション開発を行うためのフレームワークである．プロンプトとソースファイルの結合には，LangChain の提供する PromptTemplte 機能を利用した．LangChain の Models 機能を用いて，OpenAI が提供する GPT-3.5-turbo-16k をサンプリング温度 0.0 で使用した．

4.1 評 価 実 験

4. で構築した素材抽出アルゴリズムを用いて，以下のリサーチクエスチョンを確認することを目的とした評価実験を行った．

```
#instruction
これから，ターゲットソフトウェアに含まれるソースファイルを#inputとして提供します。
与えられた#contextに基づいて、これらのソースファイルからニーズに役立つのみを抽出してください。
役立つ情報が存在しない場合は"なし"と表示してください.
抽出した情報はリスト形式で示してください。

#context
開発者は以下のニーズを持っています。
needs:
{{upcycle_needs}}

これらのニーズを満たすために、ターゲットソフトウェアから関連する情報を抽出したいと考えています。
また，以下の情報は開発者が知っている知識です．抽出に役立つ場合は参考にしてください.
information:
{{related_information}}

#input: """
{{input}}
"""

#output:
```

図 1: プロンプトの雛形

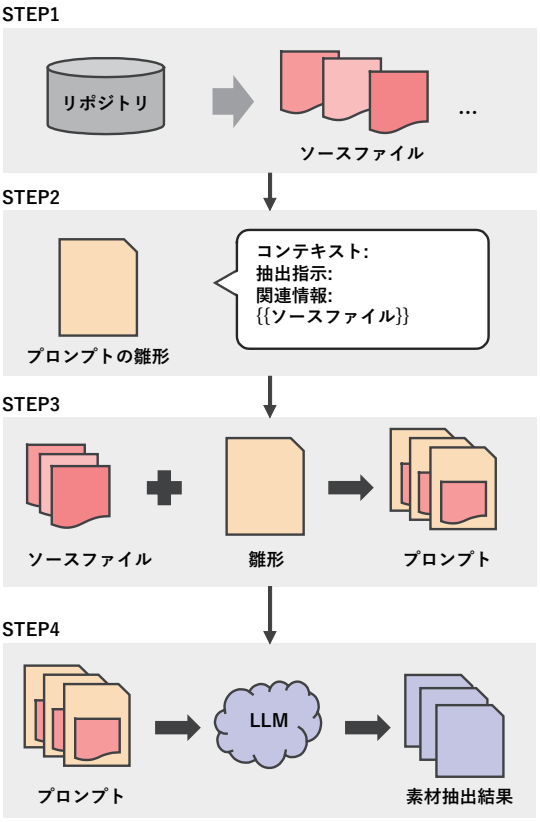

図 2: 素材抽出アルゴリズムの動作フロー

RQ1　アップサイクル素材抽出プログラムを用いて，組織内ソフトウェアから正確かつ網羅的な素材抽出は可能か？

RQ2　アップサイクル素材抽出プログラムを用いて，組織内ソフトウェアからニーズに即した素材抽出は可能か？

4.2 実 験 準 備

評価実験で使用する組織内ソフトウェアとして，神戸大学中村研究室で開発された SensorBoxManageService を使用した．SensorBoxManageService とは，複数のセンサを箱に入れて IoT として動作するようにした SensorBox [10] という物理デバイス

の管理を行う Web サービスである．SensorBoxManageService や SensorBox は神戸大学中村研究室の研究活動範囲内でのみ運用している．GPT-3.5 に対してこれらのサービスに対する情報の回答を生成させたところ，正確な情報が生成されなかったことを確認した．

評価実験では，自然言語での説明記述の影響をなくすため SensorBoxManageService のソースファイル内のコメントや仕様書を削除する．4. で作成したプログラムを用いて，SensorBoxManageService 内の計 22 件のソースファイルに対して処理を行った．評価方法については，著者がファイルごとに出力される生成結果に含まれるリスト要素の内容と，事前に作成した仕様書を照らし合わせることで RQ に関する分析を行う．

4.3 ユースケース

リサーチクエスチョンを確認するため，SensorBoxManageService を対象としたソフトウェアアップサイクルのユースケースを検討した．ユースケースごとに素材抽出に使用するプロンプトを変更し，4. で作成したプログラムを実行する．各ユースケースにおけるプロンプトの変更箇所は，3.3 で示したプロンプトの雛形における upcycle_needs と related_information である．

4.3.1 使用技術の抽出

使用される技術要素を知ることは，組織内ソフトウェアに関する知見がない場合や，全体像を掴む場合の最初のステップとして有効である．使用技術の有無は明確に判断できるため，このユースケースを通じて素材抽出の正確性や網羅的といった評価を行う．プロンプトの雛形への入力は以下のとおりである．

- upcycle_needs: このソフトウェアに用いられている技術要素 (プログラミング言語・ライブラリ・フレームワーク・API) を知りたい
- related_information: なし

4.3.2 組織内ソフトウェアに関連する実装の抽出

SensorBoxManageService では，SensorBox という研究室独自のプロダクトを扱う特性上，さまざまな組織内ソフトウェアと連携する仕組みになっている．具体的には，SensorBox の管理者や配置場所といったプロパティ情報は Scallop4SC [11] というサービスから API 経由で呼び出しており，SensorBox の計測した時系列データの可視化には TimeSeriesViewer という web サービスを利用している．こうした組織内ソフトウェアの関連する実装についての情報抽出を試みる．

プロンプトの雛形への入力は以下の通りである．

- upcycle_needs:どのようにセンサボックスに関する情報を管理しているか知りたい
- related_information: なし

4.3.3 事前知識を含めた組織内ソフトウェアに関連する実装の抽出

LLM は非公開情報やドメイン知識に関する出力を苦手とする．そこで，このユースケースでは SensorBoxManageService の管理対象となる SensorBox にまつわる情報を与える．具体的には，4.3.2 のプロンプトに加えて，related_information として図

- SensorBoxとは，複数のセンサを箱に入れてIoTとして動作するようにしたもの
  - 各ボックスには識別のためのIDがついている (例)sbox-phidget-446048
  - ボックスが収容するセンサは，センサボックス定義ファイルで自由に定義する

図 3: SensorBox に関する説明

表 1: ファイルごとの出力結果の概要

ユースケース	平均文字数	平均要素数
1	130	4
2	776	15
3	718	16

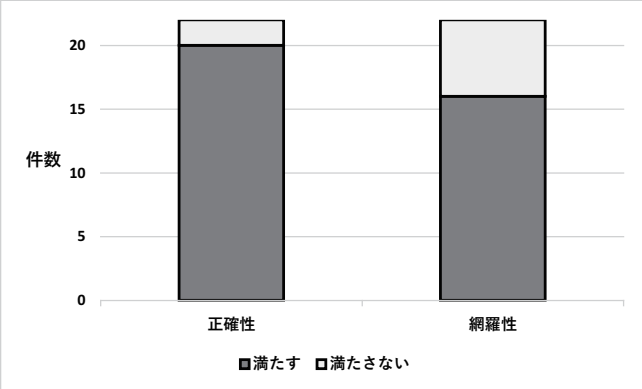

図 4: 4.3.1 の評価結果

3 の情報をプロンプトに付加する．

4.4 実験結果

各ユースケースについて，各ファイルごとの出力結果の概要を表 1 に示す．平均要素数とは，LLM からの出力に含まれるリスト要素をファイルごとに算出した際の平均値を表す．平均文字数とは，LLM からの出力に含まれる文字数をファイルごとに算出した際の平均値を表す．LLM への入出力はいずれも利用モデル（gpt-3.5-turbo-16k）のトークン上限である 16k token の範囲内であり，すべての出力結果についてリストでの出力形式をとっていた．

図 4 は，出力結果について 4.3.1 の観点から内容正確性，網羅性の２つの特性から分析したものである．内容正確性とは，入力となるソースファイルの内容から判断して，虚偽の内容が含まれていないかを示す．網羅性とは，ニーズに関する情報が出力に全て含まれているかを示す．

図 5, 6 の結果は，4.3.2, 4.3.3 の観点から出力結果について内容正確性，適切性の２つの特性から分析したものである．適切性とは，ニーズと文脈に即した出力であるかどうかを示す．例えば，出力結果がソースファイルの内容を正しく説明しているが，ニーズに即した情報でない場合は適切性を満たさないと判断する．

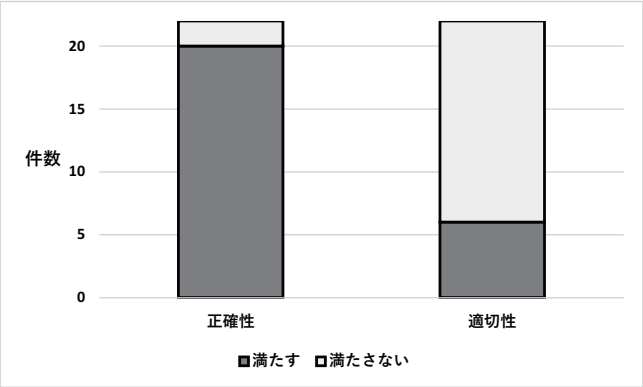

図 5: 4.3.2 の評価結果

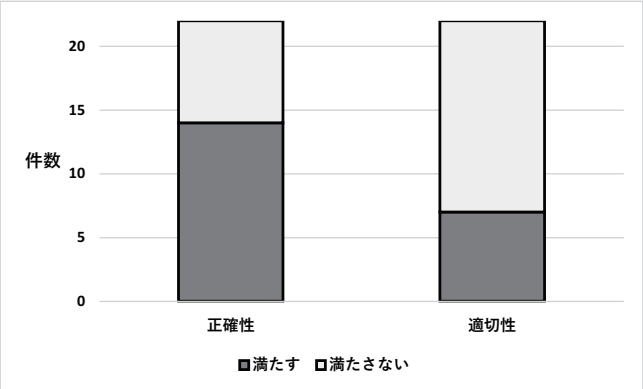

図 6: 4.3.3 の評価結果

5. 考　　察

5.1 RQ1: アップサイクル素材抽出プログラムを用いて，組織内ソフトウェアから正確かつ網羅的な情報取得は可能か？

図 4 は, 4.3.1 での技術要素の抽出における内容正確性が 90% となったことを示す．実験で用いたデータには自然言語でのコメントなどは存在しないため，ソースファイル内で用いられている記法や構文などの情報から，幅広く技術要素の抽出が可能であるとわかった．一方で，網羅性を満たさない例が 6 件存在した．これらはすべて Scallop4C といった組織内ソフトウェアの利用を抽出できなかったことが要因である．これらの組織内ソフトウェアはソースファイル内で REST API の形式で利用されており，エンドポイントに含まれる文字列のみしかそういったソフトウェアとの関連を示す情報が存在しなかった．このように，サービス名の文字列のみから組織内で用いられるライブラリ・API であることやその働きを認識することは困難である．これらの結果から，組織内ソフトウェアに用いられる一般的な技術要素の抽出は可能であるが，他の組織内ソフトウェアといった技術との関連を抽出することは困難である事がわかる．

図 5 は，4.3.2 での組織内ソフトウェアからの素材抽出における内容正確性は 90% であることを示す．いずれもソースコード内に記述された実装を正しく説明していた．正確性を満たさないと判断した 10% については内容に関連する単語の羅列のみが抽出されており，詳細な説明が省略される結果となった．

一方で，4.3.2 に加えて組織内ソフトウェアの付加情報を与えた 4.3.3 については，図 6 から内容正確性が 64% であることがわかる．付加情報に関連するソースファイルについてはより詳細に情報が抽出できたが，関連のないソースファイルに対しては図 3 で示した付加情報がそのまま抽出結果に含まれる結果となった．4.3.2, 4.3.3 における付加情報の有無が内容正確性に与える影響について述べる．4.3.3 では，ソースファイルの内容とは無関係な出力内容が多く見られた．具体的には，単にプロンプト内の情報を鸚鵡返しするような内容であり，図 5 と比較して内容正確性が 26% 低下している．

今回用いたプロンプトでは，一律にすべてのソースファイルに付加情報とともに LLM へ入力を行っていた．本研究でのプロンプトを用いてソースファイルからの抽出を行う際は，コンテキストを補強する付加情報の内容が虚偽の出力を生む大きな要因となる事がわかった．

5.2 RQ2: アップサイクル素材抽出プログラムを用いて，組織内ソフトウェアからニーズに即した情報は抽出可能か？

図 5, 6 について，適切性がそれぞれ 27%, 31% と低い結果となった．適切性を満たさないと判断した主な要因は，多くの出力内容が網羅的なソースファイルの説明となっていた点である．ソースファイル中にニーズに即した情報が存在しない場合については存在しない旨を出力するようにプロンプトを設計していたが，これらの指示通りに出力をする例は数件しか見られなかった．

4.3.2, 4.3.3 における付加情報の有無が適切性に与える影響について述べる．5.1 では，ニーズへの関連性が薄いソースファイルの出力がソースファイルへの網羅的な説明となる傾向にあることを確認したが，付加情報によって出力内容が変化する例も見られた．図 7 は，4.3.2, 4.3.3 それぞれについての SensorBoxManageService に含まれる getConfig2.cgi というソースファイルへの出力結果である．4.3.2 では詳細に実装流れを説明する内容であるのに対して，4.3.3 ではエンドポイントの管理という主題に関連のある内容に絞った情報を出力していることが分かる．プロンプトに付加情報を与える場合，ソースファイルとの関連性が薄い場合は出力の正確性を下げる可能性がある．一方で，ソースファイルの内容を補強する情報を付加した場合は適切性を上げることがわかった．

6. ま　と　め

本研究では，組織内ソフトウェアに対する，当事者の能力・知識に依存しないアップサイクル素材の抽出手法を提案した．キーアイデアとして LLM を利用したソースファイルからのアップサイクル素材抽出に着目し，具体的な実装を与えて評価実験を行った．実装では，LLM として gtp-3.5-turbo-16k を利用して筆者の所属する研究室で管理するソフトウェアに対する抽出を行った．

結果として，技術要素の抽出や実装の詳細な説明の取得には高い正確性が確認できた．また，プロンプトに付加情報を与える場合，ソースファイルとの関連性が薄い場合は出力の正確性

・対象のソフトウェアはPerlプログラミング言語を使用しています。
・ソフトウェアは以下のモジュールをインポートしています:
HTML::Template、LWP::Simple、XML::Simple、CGI,
Data::Dumper、Encode、JSON。
・ソフトウェアは定数変数`$CONFDIR`を定義し、値を`'./conf'`に
設定します。
・ソフトウェアは`CGI`モジュールの`new`メソッドを使用してCGI
オブジェクトを作成します。
・ソフトウェアはCGIオブジェクトからパラメータ`'id'`の値を取得し、
変数`$id`に割り当てます。
・パラメータ`'id'`が指定されていない場合、ソフトウェアはメッセー
ジ「デバイスIDを指定する必要があります。」と共に`error`サブ
ルーチンを呼び出します。
・ソフトウェアは引数`$id`を使用して`getSensorBoxInfo`サブ
ルーチンを呼び出し、結果を変数`$info`に割り当てます。

(a) 付加情報無し

・このソフトウェアは、ウェブサービスを使用してセンサーボックス
に関する情報を取得します。
・このソフトウェアには、サービスエンドポイントのURLとScallop
APIのURLがあります。
・このソフトウェアは、デバイスIDのパラメータを期待しています。
・このソフトウェアは、getSensorBoxInfo()関数を使用してセン
サーボックスの情報を取得します。この関数はエンドポイントに
ウェブサービスコールを行い、応答をJSONデータとして返します。
・このソフトウェアは、getLocationInfo()関数を使用して位置情報
を取得します。この関数はScallop APIにウェブサービスコール
を行い、位置情報を文字列として返します。
・このソフトウェアは、getUserInfo()関数を使用してユーザー情
報を取得します。この関数はScallop APIにウェブサービスコー
ルを行い、ユーザー情報を文字列として返します。

(b) 付加情報あり

図 7: getConfig.cgi に対する素材抽出結果の一部

を下げる可能性があることを確認した．一方で，ソースファイルの内容を補強する情報を付加した場合は適切性を上げることが分かった．将来研究として，プロンプトの指示内容やニーズの具体性が抽出結果に及ぼす影響について検討したいと考えている．

謝辞　本研究の一部は JSPS 科研費 JP19H01138, JP20H05706, JP20H04014, JP20K11059, JP22H03699, JP19K02973, 若手研究 23K17006 の助成を受けて行われている．

文　　献

[1] G. Digkas, M. Lungu, A. Chatzigeorgiou, and P. Avgeriou, “The evolution of technical debt in the apache ecosystem,” Software Architecture, eds. by A. Lopes and R. deLemos, pp.51–66, Springer International Publishing, Cham, 2017.

[2] T. Nakata, S. Chen, S. Saiki, and M. Nakamura, “Succeed: Sharing upcycling cases with context and evaluation for efficient software development,” Information, vol.14, no.9: 518, pp.••–••, Sept. 2023.

[3] 中田匠哉，陳　思楠，佐伯幸郎，中村匡秀，“ソフトウェアアップサイクルのための事例共有シ ステムの開発と評価，” 電子情報通信学会技術研究報告，第 122 巻，pp.151–156，March 2023．沖縄，名護市産業支援センター．

[4] W.X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, Y. Du, C. Yang, Y. Chen, Z. Chen, J. Jiang, R. Ren, Y. Li, X. Tang, Z. Liu, P. Liu, J.-Y. Nie, and J.-R. Wen, “A survey of large language models,” 2023.

[5] N. Nguyen and S. Nadi, “An empirical evaluation of github copilot’s code suggestions,” Proceedings of the 19th International Conference on Mining Software Repositories, p.1–5, MSR ’22, Association for Computing Machinery, New York, NY, USA, 2022. https://doi.org/10.1145/3524842.3528470

[6] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y.J. Bang, A. Madotto, and P. Fung, “Survey of hallucination in natural language generation,” ACM Comput. Surv., vol.55, no.12, pp.••–••, mar 2023. https://doi.org/10.1145/3571730

[7] “Prompt engineering guide,” `https://www.promptingguide.ai/`. (Accessed on 10/21/2023).

[8] “Best practices for prompt engineering with openai api,” `https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-openai-api`. (Accessed on 10/21/2023).

[9] “Langchain,” `https://docs.langchain.com/docs/`. (Accessed on 10/21/2023).

[10] S. Sakakibara, S. Saiki, M. Nakamura, and S. Matsumoto, “Implementing autonomous environmental sensing in smart city with IoT-based sensor box and cloud services,” Information Engineering Express (IEE), vol.4, no.1, pp.1–10, March 2018.

[11] S. YAMAMOTO, S. MATSUMOTO, S. SAIKI, and M. NAKAMURA, “Using materialized view as a service of scallop4sc for smart city application services,” Advances in Intelligent Systems and Computing, vol.271, pp.51–60, March 2014.