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
SC2024-12(2024-06)

ソフトウェアアップサイクル事例自動蓄積手法の検討

中田　匠哉† 　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: †tnakata@es4.eedept.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　ソフトウェアアップサイクルとは，既存のソフトウェア資源を再利用することでソフトウェア開発プロセスの効率化を目指す手法である．先行研究では，手入力でアップサイクルの実績を事例として共有する SUCCEED システムがある．本研究では，Stack Overflow の Q&A とアップサイクル事例の類似性に着目し，大規模言語モデル（LLM）を活用した事例自動蓄積手法を提案する．Web API を活用してデータ取得・変換・登録フローを全て自動化する．提案手法を定期実行スクリプトとして実装する．自動蓄積したデータを評価した結果，提案手法によって Stack Overflow の Q&A を適切に変換したアップサイクル事例を蓄積できることが分かった．

キーワード　ソフトウェアアップサイクル，ソフトウェア再利用，Stack Overflow，LLM，集合知

A Study of Software Upcycling Case Automatic Accumulation Method

Takuya NAKATA†, Sinan CHEN†, Sachio SAIKI††, and Masahide NAKAMURA†,†††

† Kobe University, 1–1 Rokkodai-cho, 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: †tnakata@es4.eedept.kobe-u.ac.jp, ††chensinan@gold.kobe-u.ac.jp, †††saiki.sachio@kochi-tech.ac.jp,
††††masa-n@cmds.kobe-u.ac.jp

Abstract　Software upcycling is a method aimed at enhancing the efficiency of software development processes by reusing existing software resources. Previous studies have introduced the SUCCEED system, which shares achievements in upcycling as cases through manual input. This study focuses on the similarities between Stack Overflow Q&As and upcycling cases, proposing an automated accumulation method utilizing a large-scale language model (LLM). It automates the entire flow of data acquisition, transformation, and registration using a Web API. The proposed method is implemented as a script that runs periodically. The evaluation of the automatically accumulated data demonstrates that the method can appropriately convert and store Q&As as upcycling cases.

Key words　Software upcycling, Software reuse, Stack Overflow, LLM, Collective intelligence

1. はじめに

ソフトウェア開発では，様々なプログラミングの知識やコンピュータ・情報技術に関する知識が要求される．インターネットで検索を行うことで，技術者が書いた大量の記事や Q&A などにアクセスすることができ，開発に役立てることができる．インターネットや組織内で共有されているソフトウェアリポジトリには，リリースされたソフトウェアのソースコードやドキュメントが含まれており，完成された開発知見として有用である．しかしながら，これらの知識には活用しやすくまとめられていないものや誤ったものが含まれているものがあるため，開発者が現在取り組んでいる開発に役立つ知識を探したり内容を理解したりするまでに時間がかかる場合がある．

ソフトウェアアップサイクルとは，ソフトウェアプロジェクトに含まれるソースコードやドメイン図といったプロジェクト素材を活用して，新たなソフトウェア開発を効率的に行う技術である [1]．一般的なソースコード再利用との違いは，プロジェクト素材をソースコード片レベルまで分解せずにそのまま開発に活用することで，プロジェクト素材が持つアイデアや構造などの意味的価値を利用する点である [2]．アップサイクル

は価値を高める発想が求められる再利用であるため，再利用のアイデアを思いつくことに困難さがあった．先行研究では，**SUCCEED システム**（Sharing Upcycling Cases with Context and Evaluation for Efficient Software Development システム）を提案し，ソフトウェアアップサイクルの実績やアイデアを知識ベースとして共有することで，アップサイクルの手軽な実践を推進している [3]．アップサイクルの背景・素材・レシピ・結果を主要な 4 要素としたアップサイクル事例データモデルを定義し，開発に活用しやすい形式で開発知見を扱うことを可能としている．先行研究の課題として，アップサイクル事例が手入力であるため事例の件数が集まりづらいことが挙げられる．また，技術的困難さとして，開発知見の意味を正しく理解したうえで事例の背景や結果といった要素を設定することは容易ではなく，事例生成プロセスを完全に自動化することは困難である．

本研究では，Stack Overflow の Q&A を LLM を用いてアップサイクル事例に変換し，SUCCEED システムに自動蓄積する．キーアイデアとして，Stack Overflow の Q&A とアップサイクル事例の意味的類似性に着目する．本質的に同質な構造を持つ異種データに対して大規模言語モデル（LLM）を用いてデータ変換を行うことで，意味的構造を持つデータモデルの自動生成という技術的困難さを解決する．具体的には，LLM にソフトウェアアップサイクルの定義とアップサイクル事例データモデルの定義を与えたうえで Q&A データを入力とすることで，事例データを自動生成する．さらに，Web API によって Q&A の取得を自動化し，SUCCEED システム API によって事例登録を自動化する．実装では，提案手法を定期実行可能なスクリプトとして実現する．また，提案手法を用いて自動生成した 10 件の事例について詳細に分析を行う．

2. 関 連 研 究

LLM を用いた自動データ生成・変換の研究が近年活発に行われている．Sharma らの研究では，LLM を用いてデータ変換を行う SQL コードを生成した [4]．本研究は LLM を用いて直接のデータ変換を行う点で異なる．Borisov らの研究では，テキストデータを LLM で表形式に変換するアプローチを提案した [5]．テキストの意味を解釈して表の各要素にデータを当てはめる点で本研究と類似しているが，本研究はアップサイクルを説明する長文のテキスト出力やアップサイクル素材の配列の可変長配列といった特殊な条件でのデータ変換である．Peng らの研究では，農業に関する文章から LLM で害虫駆除に関する構造化データを抽出した [6]．文章から指定のデータ構造を抽出するという点で本研究と類似しているが，本研究は必要なデータのみをピックアップするのではなく文章全体の知見を総合的にデータとして抽出する．

3. 準　　備

3.1 ソフトウェアアップサイクル

「車輪の再発明」とは，開発したソフトウェアの一部またはほぼ全てが，一般的に普及した製品・ライブラリで十分に代用できたことが後から発覚することである．学習目的には有用だが，実際の開発ではコスト増大につながる．設計段階で技術調査を十分に行うことで防止できる可能性があるが，代用技術に気付かず再発明してしまうことも多い．ソフトウェアアップサイクルとは，既存のソフトウェア開発の知見を発掘し新たな開発に役立てる技術である．ソースコード再利用を発展させ，開発知見を幅広く素材として再利用する．すなわち，開発に使える素材としての「車輪」を増やす試みである．使われないが有用な知見をそのまま開発に活用することで付加価値サービスとして生まれ変わらせる．ソフトウェアアップサイクルの特徴としては，素材に使える開発知見の種類の多さがある．ソースコード片・ドメイン・アーキテクチャ・テストケースといったコード関連のものから，頻繁に発生する不具合・不具合の解決方法・技術的 TIPS といった便利な知識や，ソフトウェアそのもの・ソフトウェアの根幹にあるアイデアといった根本的なものまで，ソフトウェアに関連する知識であれば種類を限定しない．アップサイクルの技術的課題としては，知見の発掘・活用が容易ではないことが挙げられる．

3.2 インターネット上の開発知見

インターネット上には手軽に利用できる開発知見が多く存在し，実際の開発でも活用されることが多い．Stack Overflow は，情報技術に関する質問と回答を共有する英語サイトである [7]．2008 年にサービスを開始し，2024 年 5 月時点でユーザ数は 2400 万人である．プログラミング技術を中心としたトピックの幅広さが特色である．質問者がベストアンサーを選択したり，閲覧者が投稿に高評価を付ける機能がある．ユーザによる全ての記述が CC-BY-SA ライセンスであり，適切な記述のもと記述を改変しての利用が可能となっている [8]．その他にも技術記事投稿サイトが利用されることが多い．日本国内では Qiita や Zenn などのサービスがある [9], [10]．閲覧者が投稿に高評価を付ける機能がある．CC ライセンスではなく，記事そのものの利用や改変は著作権で保護されている．ソフトウェアリポジトリ共有サイトの GitHub も開発知見として有用である [11]．既にリリースされた大小さまざまなプロジェクトのソースコード・アーキテクチャを閲覧することができる．しかしながら，ソースコードファイルがそのまま置かれておりドキュメントとしてまとめられていないため，内容の読み込みには専門知識が必要だったり欲しい知識にたどりつくまで時間がかかることがある．リポジトリごとに設定されているライセンスが異なり，CC ライセンスのものも多くある．

3.3 先行研究：SUCCEED システム [3]

SUCCEED システム（Sharing Upcycling Cases with Context and Evaluation for Efficient Software Development システム）は，ソフトウェアアップサイクルの事例を共有する知識ベースシステムである．共有された具体的なアップサイクルの実績を参考にして開発を行うことで，手軽にアップサイクルを行うことができるシステムである．アップサイクル事例のデータモデルが定義されており，モデルの各属性は基本的に自然言語により自由記述された文字列で表現される．事例データモデルのコアとなる 4 つの属性を以下に示す．

- 背景（`context`）：開発の目的・経緯・課題

- 素材（materialList）：開発に役立った素材の一覧
- レシピ（recipe）：素材の活用方法
- 結果（result）：開発の結果良かった点や悪かった点．

合わせて 5 点満点の評価もつける．

組織内にデータベースを構築し，アップサイクル事例を登録・共有する．各自が見つけた役立つソースコードや技術をアップサイクル事例にまとめることで，開発に活用しやすい形式で共有することができる．キーワードを用いた事例の検索が可能である．学生を対象とした実証実験において，SUCCEED システムを用いた開発の効率化が示された．SUCCEED システムの課題は，開発効率化のためには事例を大量に登録する必要があることである．手作業での事例登録には手間がかかり，短時間での大量の登録は現実的ではない．しかしながら，任意の開発知見を意味的に解析してアップサイクル事例データ構造に変換することは容易ではない．本研究では，LLM を用いて意味的解析の困難性という課題を解決することで事例の自動登録技術を確立し，事例の蓄積件数を増加させることを通じて SUCCEED システムによる開発支援の効果を上昇させる．

4. 提 案 手 法

4.1 目的とキーアイデア

本研究の目的は，Stack Overflow の Q&A をアップサイクル事例として SUCCEED システムに自動蓄積する手法を確立することである．キーアイデアとして，Stack Overflow の Q&A とアップサイクル事例の間にある本質的な類似性に着目した．Q&A における困りごとはアップサイクルの背景に該当し，質問と回答に記載されたソースコードやライブラリの URL はアップサイクルの素材に該当する．回答には素材を用いたアップサイクルのレシピが記載されており，質問者と回答者が想定した結果になったことがベストアンサーに選ばれたことによってある程度保証されている．このことから，LLM を使った文字列変換によって，非構造的な自然言語で書かれた Q&A を構造的なアップサイクル事例に自動変換し，これを SUCCEED システムに登録することで，アップサイクル事例の自動蓄積が実現できると考えられる．本研究は以下の 3 つのアプローチで行う．

（A1） Web API による開発知見の取得
（A2） LLM によるデータ形式の変換
（A3） SUCCEED システムへの事例登録

提案するアップサイクル事例自動蓄積手法は，図 1 に示すフロー図に従って動作する．各ステップがアプローチ（A1）から（A3）に対応しており，詳細な処理を順に説明する．

4.2 （A1）Web API による開発知見の取得

Stack Overflow からの開発知見の取得を行う．Stack Overflow の Web API である Stack Exchange API v2.3 を使用する [12]．この API の特徴として，検索クエリの柔軟性が挙げられる．Stack Overflow の Q&A の各データは膨大な種類の要素を持つが，全ての属性について API で取得するかどうかを個別に選択することができる．検索クエリを活用して必要な要素を追加したり不要な要素を削除することで，Stack Overflow 上のデータを扱いやすくする．API を用いたデータ取得とデータ整形の流れは以

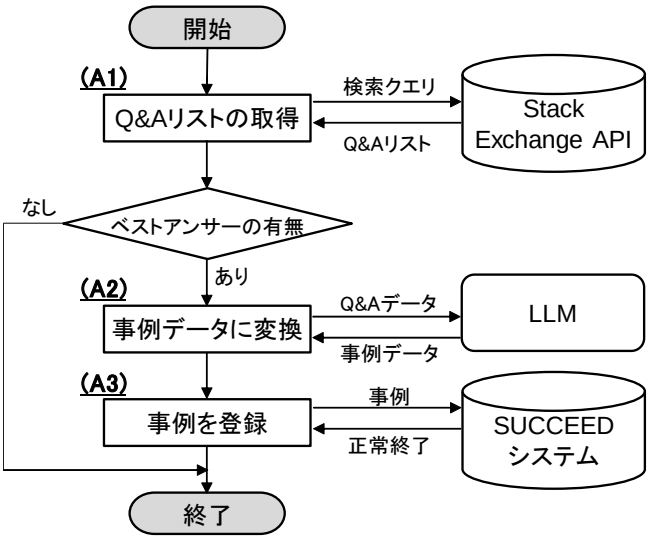

図 1: アップサイクル事例自動蓄積フロー

下のとおりである．

（1） 検索クエリによる質問と回答の一覧の取得
（2） ベストアンサーが存在する質問のみの取得
（3） 質問とベストアンサーのデータ整形

まず，検索クエリによって必要なデータ要素のみを選択し，人気の高い質問の一覧を人気度降順で取得する．これにより，有用な開発知見であると期待できる評価の高い Q&A のペアを取得する．API では，直近・一週間・一か月などの期間を選択し，その期間で人気の高い Q&A を取得することができる．検索クエリで取得した質問の属性は，質問の投稿者・質問の本文（マークダウン形式）・質問のスコア・質問のタイトル・質問ページのリンク・accepted_answer_id・質問に紐づく回答の一覧である．検索クエリで取得した各回答の属性は，回答の投稿者・回答の本文（マークダウン形式）・回答のスコア・is_accepted である．

次に，質問の一覧の中からベストアンサーが存在する質問のみを取り出す．適切な回答が存在しない質問には，アップサイクル事例における素材・レシピ・結果に該当する内容が欠落している可能性があるため除外する必要がある．Stack Overflow には，質問に対する回答の一覧の中から，質問の投稿者がベストアンサーを選択する機能が存在する．accepted_answer_id 要素は，ベストアンサーが選択されている質問のみが持ち，ベストアンサーの回答 ID を表す．すなわち，accepted_answer_id 要素を持つ質問のみを取り出すことで，ベストアンサーが存在する回答を取得することができる．

最後に，質問とベストアンサーから必要なデータを抜き出して整形する．ベストアンサーは回答の一覧の中で is_accepted が真となっている唯一の回答である．最終的に抜き出した要素は，質問の投稿者・質問の本文（マークダウン形式）・質問のスコア・質問のタイトル・質問ページのリンク・回答の投稿者・回答の本文（マークダウン形式）・回答のスコアである．これらの要素を質問ごとに取得することができる．

```
- title
- question
- answer
```

図 2: LLM への入力の Q&A データモデル

```
- title
- context
- materialList（配列形式）
  - detail
  - access
  - version
  - discovery
- recipe
- resultDetail
```

図 3: LLM からの出力の事例データモデル

4.3 （A2）LLM によるデータ形式の変換

LLM を用いて（A1）で取得した開発知見をアップサイクル事例に変換する．LLM に入力としてプロンプトと Q&A 形式のデータを与えると，出力としてアップサイクル事例形式のデータが得られる．教師データを用いないゼロショットプロンプティングで実行する．プロンプトの構造は以下のとおりである．

（1） ソフトウェアアップサイクルとは何かの説明
（2） アップサイクル事例の説明
（3） アップサイクル事例のデータ構造定義
（4） 入力に基づいて事例の抽出を依頼
（5） 結果を日本語に指定
（6） 結果を JSON 形式に指定

プロンプトに続いて LLM に入力するデータ構造を図 2 に，LLM からの出力のデータ構造を図 3 に示す．LLM への入力は，（A1）で取得した Stack Overflow の開発知見をもとに作成した Q&A データモデルである．入力の要素は，タイトル・質問の本文・回答の本文である．本文はどちらもマークダウン形式である．マークダウン形式の特徴として，文章以外に文章構造を表現するための記号が文頭などに含まれている一方で，HTML 形式などのタグを多用するマークダウン形式に比べると記号の量が少ない．API から取得できる本文データはマークダウン形式と HTML 形式の 2 種類があるが、より記号が少ないマークダウン形式の方が LLM への入力に適した形式であると考えられる．また，本文に使用されている言語は英語である．出力の要素は，タイトル・コンテキスト・アップサイクル素材リスト・レシピ・結果詳細である．出力の形式は JSON 形式であり，各要素は日本語で記述される．素材リストは入れ子になった JSON 形式の配列であり，各 JSON データが一つのアップサイクル素材に対応する．アップサイクル素材の要素は，詳細・アクセス方法・バージョン・素材を見つけた経緯である．

4.4 （A3）SUCCEED システムへの事例登録

まず，（A2）で変換したデータに加えて，SUCCEED システムへの登録に必要な `creator` と `resultScore` の 2 つを付加する．`creator` は，事例の作成者である．Q&A では，質問と回答で作成者が異なるため，両方の作成者のユーザ名を並べて記述する．`resultScore` は，事例の主観的評価を 5 段階で評価した点数で，点数が高いほど高評価となる．Stack Overflow では，各投稿ごとに高評価数を記録しており，質問と回答で別に記録されている．そのため，質問と回答の高評価数を参考にして `resultScore` の算出が可能である．`resultScore` は主観的な数値であるため評価には曖昧さが伴うが，例えば質問と回答の高評価数がともに 10 件以上だと 5 点とする，といった点数の付け方が考えられる．

次に，提案手法によって手入力ではなく外部ソースにもとづく事例が SUCCEED システムに登録されるようになったことから，事例データモデルに情報源を表す要素を追加する必要が生じる．そのため，`source` と `sourceLink` と `license` の 3 つの要素を追加する．`source` は，情報源の種類を表すもので，手入力であれば NULL 値を，Web サイトであれば Web サイト名を入力する．本提案では Stack Overflow が情報源であるため，`source` は “Stack Overflow” となる．`sourceLink` は，情報源の URL である．本提案では事例の元となった質問の Stack Overflow ページの URL となる．`license` は，事例のライセンスである．表記がない場合はライセンスなしとなり，著作権法で保護される範囲で無断での利用・改変・再配布等ができない．Stack Overflow では，近年投稿された記事に対して CC-BY-SA 4.0 ライセンスが付与されている．SA ライセンスは加工後のコンテンツに対しても同一のライセンスが付与されるため，Q&A を加工して使用しているアップサイクル事例に対しても CC-BY-SA 4.0 ライセンスが付与されると考えられる．そのため，`license` は “CC-BY-SA 4.0” となる．

（A2）と（A3）で生成したアップサイクル事例の要素は，`title`, `context`, `materialList`, `recipe`, `resultDetail`, `resultScore`, `creator`, `source`, `sourceLink`, `license` の 10 要素である．これらの要素を SUCCEED システムが提供するアップサイクル事例登録 API を用いて，SUCCEED システムに登録する．これによって，Q&A を取得してデータ変換しアップサイクル事例としてデータベースに蓄積するまでの流れを自動化できる．

5. 実　　装

提案手法を Python を用いた一連の処理として実装した．（A1）におけるデータ整形が複雑であるため，Python におけるデータ分析ライブラリの Pandas を使用した．

LLM は OpenAI が提供する chatgpt-3.5-turbo を利用した．サンプリング温度を 0.0 に設定することで LLM の出力を安定させ，LLM によって Q&A に記述されていない不確定な要素が追加されにくくした．LLM への入力プロンプトを図 4 に示す．（A2）で示したプロンプトの構造に従って英語で記述した．`materialList` の要素のうち，`access`・`version`・`discovery` が頻繁に欠落するため，事例登録時のエラーを防ぐ対策として欠落時のデータを “None” に統一する工夫を行った．OpenAI API の response_format 機能を用いて，LLM の出力を JSON 形

```
Software upcycling is a type of software reuse that leverages insights from software development and materials such as source code to enhance the overall efficiency of software development.
A software upcycling case is a structured description of an instance where software upcycling was implemented.
It includes the following structure:

- title
- context: The background and reasons for implementing upcycling.
- materialList: A list of materials used in the upcycling.
  - Structure of each material: (If it is null, it is set to "None" not to null.)
    - detail: A summary of the material.
    - access: The URL of the material.
    - version: The semantic version of the material.
    - discovery: How the material was discovered.
- recipe: How the materials were processed to perform the upcycling.
- resultDetail: The outcomes of the upcycling, both positive and negative.

From the following technical Q&A, please extract a case of software upcycling.
The language must be in Japanese.
Output should be in JSON format following the structure of a software upcycling case.
```

図 4: 実装で用いた LLM の入力プロンプト

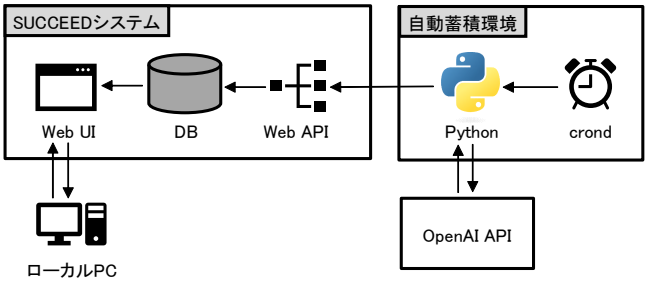

図 5: 実装アーキテクチャ

式に指定した．この機能を使うことで，LLM の出力フォーマットが JSON 形式に従うことが保証される．

提案手法を実行するスクリプトは，SUCCEED システムとは別のサーバー上で動作させた．Linux のスクリプト的実行ツールの cron を用いて，週に 1 回スクリプトを実行した．（A1）で Q&A がどの期間で人気かを選択できたが，実装では直近一週間で人気の Q&A を 20 件取得する検索クエリを用いた．取得した回答のうち，ベストアンサーが存在する回答のみを事例に変換して SUCCEED システムに蓄積した．このように，アップサイクル事例の自動蓄積を定期的に実行する仕組みとして提案手法を実装した．実装アーキテクチャを図 5 に示す．

また，本提案で事例データモデルに追加した `source`・`sourceLink`・`license` の 3 要素を表示する機能を SUCCEED システムの Web ユーザインタフェース（Web UI）に追加した．Q&A の回答の投稿者と質問の投稿者も `creator` として登録したことによって Web UI 上に表示されることが確認できた．Web UI の修正箇所を図 6 に示す．

6. 評　　価

提案手法で生成した 10 件の事例を評価した．Stack Overflow の元の Q&A と，生成した事例を比較・分析した．専門的な内容であるため，文字列比較ではなく著者が目視で内容を読んで比較を行った．結果を表 1 に示す．言語は事例が対象としているプログラミング言語，素材数は事例が含む素材の数，スコアは結果の 5 段階評価，有用度はアップサイクル事例として有用かどうかである．事例で使われた言語は，R 言語，Java，C#，

図 6: SUCCEED システムの Web UI の改善

表 1: 10 件の事例の評価

概要	言語	素材数	スコア	有用度
計算速度の向上	R 言語	1	4	高
最短経路問題	なし	1	4	高
手動コンパイル	Java	2	4	高
連続区間のグループ化	R 言語	1	4	高
最新のバグ報告	C#	3	5	低
最新のバグ報告	Python	1	5	低
コンパイラの改善	C 言語	3	5	高
サフィックスの挙動	C 言語	2	5	高
特殊な演算子定義	C++	1	5	低
パイプ実行	R 言語	3	5	高

Python，C，C++，Rust であった．

まず，Q&A に書いてある内容と生成された事例の内容に大きな違いがあるかを確認することで，ハルシネーションが発生しているかを調査した．ハルシネーションとは，LLM による意味的に誤った文章の生成である [13]．比較の結果，内容に大きな違いは見られなかったため，重大なハルシネーションは発生していないと考えられる．しかしながら，R 言語に関する事例において，事例で使用したライブラリの URL が Q&A の文中に存在しないにもかかわらず，`accesss` に適切なリンクが 4 つ生成されていた．その事例においては，`version` の情報が少し古く最新ではなかったことから，LLM が事前学習したデータの中に含まれていたものを生成したと考えられる．バージョンは最新ではないが誤りではなかったため，ハルシネーションではあるが重大な欠陥ではないと考えられる．

次に，生成された内容がアップサイクル事例としてふさわしいかという観点で評価した．最新のバグ報告や仕様変更に関する Q&A が 3 件あり，これらは知見が活用できる範囲・期間が限定的で再利用しづらい内容だったため，有用度を低と評価した．その他の 7 件は開発に役立つライブラリや改良点などの開発に有用な知見であり，アップサイクル事例として活用できる可能性があった．SUCCEED システム上の知識の幅を増やすという点においては，生成した事例の 70%が有用という結果は十分に高いと言える．しかしながら，有用度を低としたバグ報告や仕様変更の事例のスコアが全て 5 点となっていた．これは，Stack Overflow 上での高評価とアップサイクル事例としての評価が必ずしも一致しないことを示す．

また，事例に含まれる合計 18 件の素材についても分析を行っ

た．うち半分の 9 件の素材が version の値を持ち，残りの 9 件の素材では NULL であった．セマンティックバージョニングが行われているライブラリやソフトウェアを素材とした場合は，バージョンの数値を version の値としていた．GitHub 上のファイルを素材とした場合は，Git におけるバージョン管理情報であるコミット ID を自動で取得し version の値としていた．素材へのアクセス方法を示す access では，Q&A 中に直接記載されたアドバイスやソースコードを参照にする場合を除いた 15 件全てで適切な URL が記載されていた．ハルシネーションの調査でも述べた通り，Q&A 中に URL が記載されていない場合は適切な URL を LLM が生成して記載していた．発見方法を示す discovery の項目では，14 件の素材が Q&A の質問と回答のどちらに記載されていた素材であるかが記載されていた．3 件がソースコード中でのライブラリの使用意図が記載されており，1 件が回答者が Python のソースコードを直接調査した旨が記載されていた．

7. 考　　察

提案手法の利点を述べる．まず，Stack Overflow で多くの人が閲覧している Q&A を SUCCEED システム上で利用できるようになる．これにより，SUCCEED システムに蓄積される事例数が増加し，SUCCEED システムを用いた開発効率化の効果上昇が期待できる．また，評価した 10 件の事例に関しては，ハルシネーションが少なく正確なデータ変換ができたことが確認できた．さらに，英語を日本語に翻訳して蓄積することで，英語が苦手な人でも開発知見を利用できるようになるという副次的メリットもある．本提案手法によって，Q&A とアップサイクル事例のように異なるデータ構造であっても，意味的なつながり・共通点を見いだすことができれば LLM によるデータ変換がある程度正確にできる可能性があることがわかった．データ構造の本質的類似性のアイデアは，事例の生成に限らず LLM を用いた様々なデータ変換に活用できると考えられる．

提案手法の限界を述べる．まず，開発知見のソースである Stack Overflow の Q&A の内容は，必ずしも正確とは限らない．さらに，LLM を用いた変換によって，本当に元の Q&A と同じ内容の事例が生成されたかどうか確認することが困難である．本実験の評価ではハルシネーションがほぼ起こっていなかったが，他の Q&A を入力とした場合や LLM のモデルが変化した場合の品質を保証する手法が存在しない．このように，提案手法で蓄積した知見には，ソースの正確さとデータ変換の正確さという二重の正確さの壁がある．解決のアイデアとしては，正確さを評価して事例のスコアに反映する方法や，より高性能な LLM を用いて変換の正確さを上げる方法がある．また，評価によってそもそもアップサイクル事例として活用しづらい Q&A が存在することも分かった．

8. ま と め

本研究では，ソフトウェア開発に役立つアップサイクル事例をインターネット上の開発知見から自動生成することを目的として，Stack Overflow の Q&A 投稿を対象として LLM によるデータ変換を行った．API による開発知見の取得と SUCCEED システムへの事例蓄積も自動化し，アップサイクル事例の蓄積フロー全体を自動化した．実装では，cron を用いて自動蓄積フローの定期実行を行った．提案手法を用いて生成した 10 件の事例について内容を分析し，正確な内容でデータ変換が行われていることを確認した．本研究によって，異なるデータ構造間であっても意味的な類似性を見出せるものであれば，LLM によるデータ変換が可能であるという新たな仮説が生まれた．このアイデアはソフトウェアアップサイクルの研究に留まらず様々なデータ生成で活用できる可能性がある．本研究の今後の展望として，今回取り組まなかったソフトウェアリポジトリをもとにしたアップサイクル事例の自動生成手法の開発に取り組む．インターネット上の開発知見だけでなく組織内でクローズに所有するリポジトリも対象とするなど，SUCCEED システムに蓄積する事例の種類を増やしていきたい．

謝辞　本研究の一部は JSPS 科研費 JP20H05706, JP22H03699, JP22K19653, JP23H03401, JP23H03694, JP23K17006，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．

文　　献

[1] K. Terakawa, S. Chen, S. Saiki, and M. Nakamura, “A study of project description inference using method name elements for software upcycling,” 6th International Conference on Signal Processing and Information Security (ICSPIS), pp.46–51, Nov. 2023. Dubai.

[2] B. Reid, M. d’Amorim, M. Wagner, and C. Treude, “Ncq: Code reuse support for node.js developers,” IEEE Transactions on Software Engineering, vol.49, no.5, pp.3205–3225, 2023.

[3] 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, p.518, Sept. 2023. doi.org/10.3390/info14090518.

[4] A. Sharma, X. Li, H. Guan, G. Sun, L. Zhang, L. Wang, K. Wu, L. Cao, E. Zhu, A. Sim, T. Wu, and J. Zou, “Automatic data transformation using large language model - an experimental study on building energy data,” 2023 IEEE International Conference on Big Data (BigData), pp.1824–1834, 2023.

[5] V. Borisov, K. Seßler, T. Leemann, M. Pawelczyk, and G. Kasneci, “Language models are realistic tabular data generators,” 2022.

[6] R. Peng, K. Liu, P. Yang, Z. Yuan, and S. Li, “Embedding-based retrieval with llm for effective agriculture information extracting from unstructured data,” 2023.

[7] “Stack overflow - where developers learn, share, & build careers,” https://stackoverflow.com/. (Accessed on 05/15/2024).

[8] “Cc by-sa 4.0 deed — attribution-sharealike 4.0 international — creative commons,” https://creativecommons.org/licenses/by-sa/4.0/deed.en. (Accessed on 05/15/2024).

[9] “Qiita,” https://qiita.com/. (Accessed on 05/15/2024).

[10] “Zenn ｜エンジニアのための情報共有コミュニティ,” https://zenn.dev/. (Accessed on 05/15/2024).

[11] “Github,” https://github.com/. (Accessed on 05/15/2024).

[12] “Stack exchange api,” https://api.stackexchange.com/docs. (Accessed on 05/15/2024).

[13] J. Li, X. Cheng, X. Zhao, J.-Y. Nie, and J.-R. Wen, “HaluEval: A large-scale hallucination evaluation benchmark for large language models,” Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pp.6449–6464, Dec. 2023. https://aclanthology.org/2023.emnlp-main.397