



ソフトウェアアップサイクル実現のための 既存資源活用手法に関する研究

神戸大学

○寺川航平, 陳 思楠, 佐伯 幸郎, 中村 匡秀



背景: ソフトウェアの再利用

■ ソフトウェアを取り巻く環境は常に変化

- ◆ 利用者・利用形態の変動
- ◆ 開発担当者への入れ替わり
- ◆ 技術進歩

- 対応に要する人・時間等のリソースは有限
- 開発コストを削減したい



■ 開発組織には過去の開発経験が蓄積されている

◆ 組織内リポジトリに多くの価値が存在

- 組織の内情にかかわるため、外部に公開できないソフトウェア
- 組織ドメインに特化した知見・アイデア

- ソフトウェアの**アップサイクル**を検討



背景: ソフトウェアアップサイクル

- ソフトウェアプロジェクトが持つ具体的な構造や設計思想, 理念などを抽象化し, 別のソフトウェア開発に転用すること
 - ◆ 組織において放置されがちな既存ソフトウェアに着目
 - ◆ 既存資源の再利用(=アップサイクル)による開発運用の効率化を目指す

■ 例: SensorBoxManageService

- ◆ SensorBox (IoTデバイス)を管理するWebサービス
- ◆ 中村研究室で開発・運用されるサービス
 - **管理者しか**詳細を知らない・**説明文書が不十分**
 - 一方で, SensorBoxを扱う処理・考え方は利用価値がある



SensorBox

⇒ SensorBoxManageServiceは資源として**アップサイクル**可能





研究の範囲と課題

■ 研究の範囲

◆ 1. アップサイクル素材の抽出

- アップサイクル対象を理解し、素材を見つけ出す

◆ 2. アップサイクル素材の組み合わせ

■ 素材抽出における問題

◆ 必ずしも抽出対象のソフトウェアドキュメントが存在するとは限らない

- 開発時の優先度: 機能改修・新規開発 > ドキュメンテーション

◆ ソースファイルを読み解く際に多大なコストが生じる

- 速度や精度は個人の知識・能力に依存

課題: **ドキュメントの有無に関わらず、ソースファイルのみから効率的に特徴を把握すること**



研究の目的と成果

■ 目的

- ◆ 組織内における過去のソフトウェアプロジェクトからソフトウェアアップサイクル可能な資源を抽出すること

■ アプローチ

- ◆ メソッド名構成単語を用いたソフトウェア抽象化
- ◆ LLMを活用したソフトウェア抽象化

■ 本研究における取り組み

- ◆ A1: メソッド名構成単語に基づく既存プロジェクト概要推測の検討と評価
- ◆ A2: LLMを用いた単一ソースファイルからの素材抽出と評価
- ◆ A3: LLMを用いたソフトウェア全体からの素材抽出と評価



A1: メソッド名構成単語を用いた抽象化

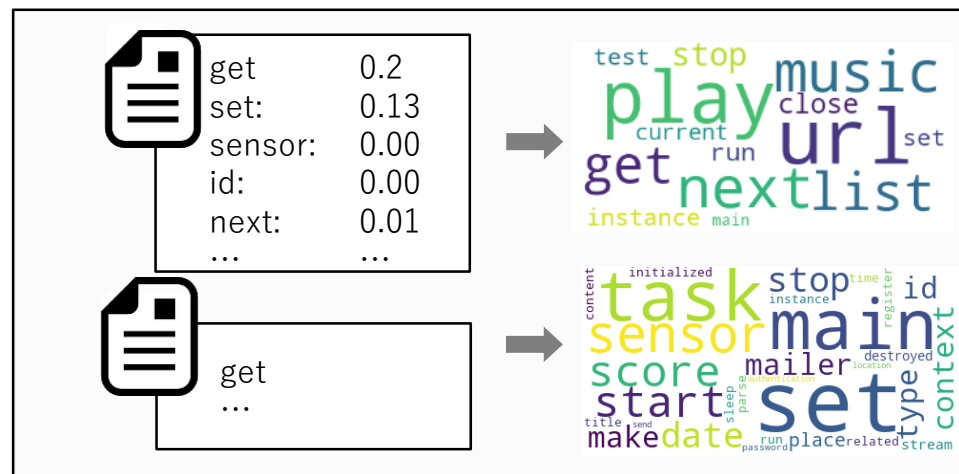
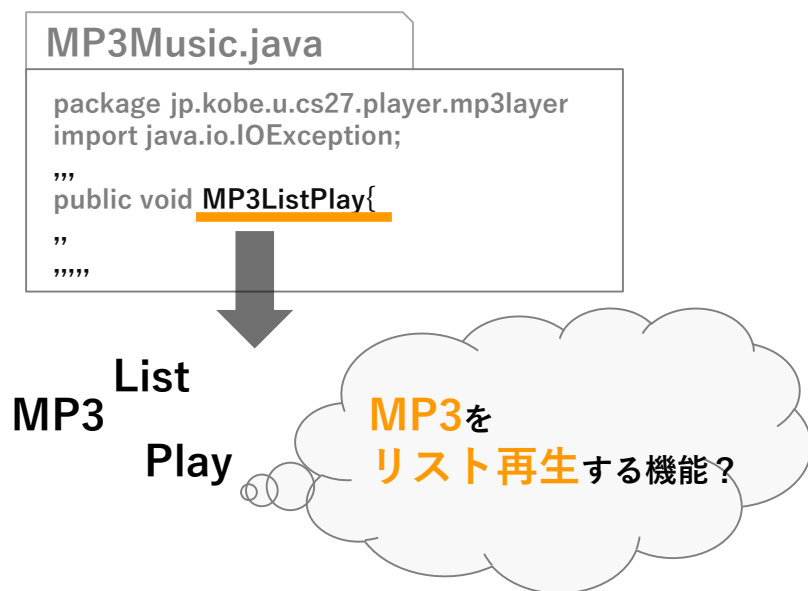
■ キーアイデア

◆ メソッド名に着目した単語抽出

- メソッド名はソフトウェアの機能に深く関連

◆ 単語の重要度による重み付け

- ソフトウェア固有の情報をtf-idfにより重みづけし、可視化する



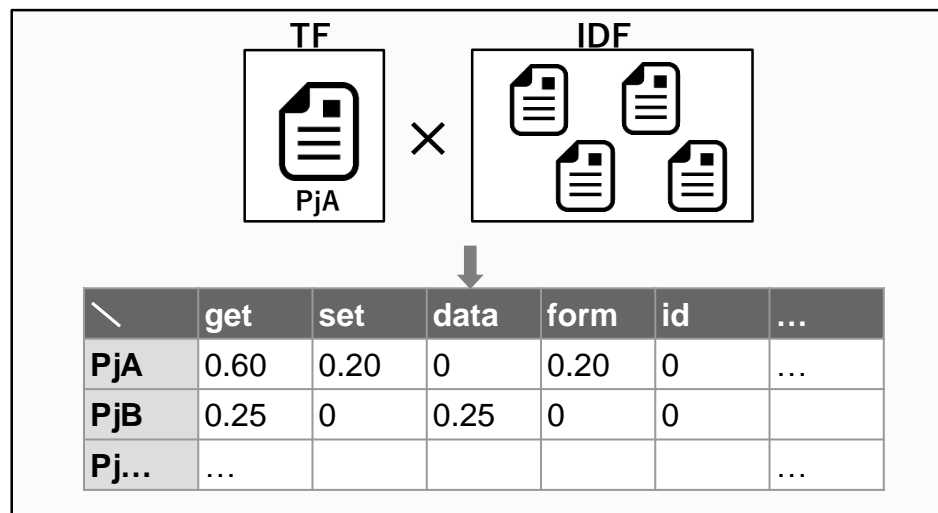
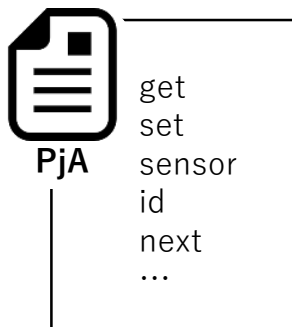


A1: tf-idfによる重み付け

■ 取得したメソッド名構成単語群に重みを設定する

- ◆ tf-idf: 各文書中に含まれる各単語が「その文書内でどれくらい重要か」を表す尺度
- ◆ ソフトウェアプロジェクトを文書とみなし, tf-idfを算出
 - tf: プロジェクト単位で見たときの単語の出現頻度
 - idf: 複数プロジェクト全体で見たときの, 単語の出現頻度
- ◆ プロジェクトの特徴を反映したベクトルを生成する

メソッド名構成単語群





取得した単語群と生成画像

■ Tag cloud画像

- ◆ データに紐付けられた重みを視覚的に表現する手法
- ◆ 重要度の大きな単語ほど大きく表示する

取得した単語群

tagcloud画像

get stream score main context destroyed context
initialized get instance start mailer task stop all mailer
task register tasks main start stop set related tasks
start sleep stop run parse date time make title make
content get password authentication send main get
location main get date set date get score set score get
sensor id set sensor id get sensor type set sensor type
get place set place





A1: 結果と考察

■ 結果

◆ 名詞

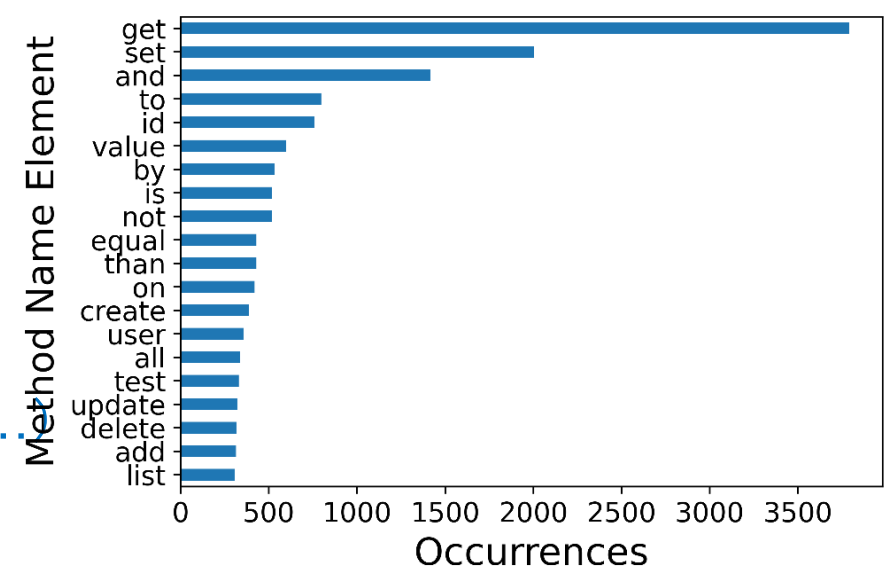
- システムが扱うモノ・コトを指す

◆ 動詞 (get, set...)

- 頻出の単語が存在

◆ 接続詞, 副詞 (and, is, not, than,...)

- 分割により意味を失う



■ 考察: LifeActivityMailer

◆ センサー値を取得, メール送信するサービス

- 図2では, **ソフトウェアの動作に関わる単語**(score, sensor)に重みがついた



A2: LLMを用いた単一ソースファイルからの素材抽出と評価

■ 目的

- ◆ ソースファイル中からソフトウェアアップサイクル素材を抽出すること

■ キーアイデア

- ◆ 近年利用が進むLLM(大規模言語モデル)に着目
- ◆ 出入力として扱えるテキストの長さには上限がある
 - すべてのソースファイルを入力することはできない

⇒ **単一ソースファイル**を入力とする

■ アプローチ

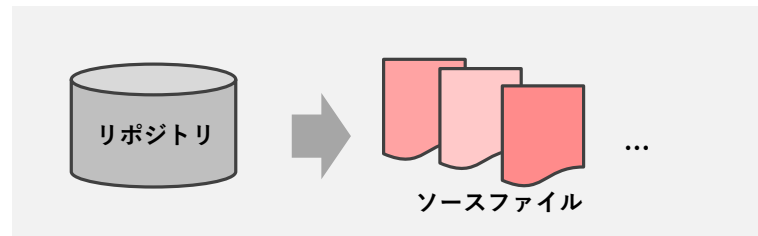
- ◆ Step1: リポジトリマイニング
- ◆ Step2: プロンプトの雛形の定義
- ◆ Step3: プロンプトの作成
- ◆ Step4: ソースファイルごとのLLM実行



Step1~2の概要

■ Step1: リポジトリマイニング

- ◆ 対象となるリポジトリから
ソースファイルをすべて取得



■ Step2: プロンプトのひな形の定義

◆ 基本的な構成要素を定める

- 命令: 素材となる情報の抽出
- 文脈: アップサイクルの経緯
- 入力データ: 単一のソースファイル
- 出力指示子: リスト形式の出力を指示

◆ プログラム実行時に変更できる箇所を用意

- アップサイクルのユースケースに応じて
実行時に値をはめ込む

```
#instruction
...
#context
{{related_information}}

#input:
{{input}}
...
#output:
```



Step3: プロンプトの作成

#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情報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つ**結合**ます。

needs:
{{upcycle_needs}}

アップサイクル時の要望
例: このコード中の使用技術が知りたい

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。

#input:

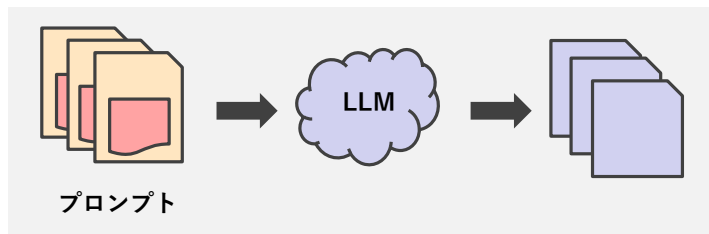
""
{{input}}
""

#output:

ソースファイル

Step4:ソースファイルごとのLLM実行

- 作成したプロンプトをLLMに入力し, 出力を得る
 - ◆ ソースファイルごとに実行する



- 出力の例: SensorBoxManageService
 - ◆ 使用したLLM: GPT-3.5 Turbo

・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`サブルーチン呼び出しします。
・ソフトウェアは引数`\$id`を使用して`getSensorBoxInfo`サブルーチン呼び出し、結果を変数`\$info`に割り当てます。



評価実験

- 抽出対象: SensorBoxManageService
 - ◆ SensorBoxというIoTを管理するWebサービス
 - SensorBoxの登録・検索といった管理機能
- 使用したモデル
 - ◆ OpenAI API: GPT-3.5 Turbo
- 評価
 - ◆ 抽出結果とソフトウェアのREADMEを照らし合わせ主観的に評価
- アップサイクルのコンテキストを検討
 - ◆ Usecase1: 使用技術の抽出
 - LLMの学習済みの技術領域について検証
 - ◆ Usecase2: 組織内ソフトウェアに関連する実装の抽出
 - 学習データに含まれない情報について検証



SensorBox

UC1:使

- アップ
- ◆ 用い

#context

開発者は以下のニーズを持っています。

needs:

このソフトウェアに用いられている技術要素が知りたい

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持つ情報です。これを踏まえて抽出を行ってください

information:

なし

■ 使用したプロンプト

#instruction

あるソフトウェアのソースファイルを#input以下に示します。

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沿った情報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

ソフトウェア開発者は、以下のニーズを持っています。

needs:

このソフトウェアに用いられている技術要素が知りたい

#input “”

{{input}}

#output



UC1の結果の一部

```
getConfig.cgi
- プログラミング言語:
  - Perl

- ライブラリ:
  - HTML::Template
  - LWP::Simple
  - XML::Simple
  - CGI
  - Data::Dumper
  - Encode
```




UC1:結果と考察

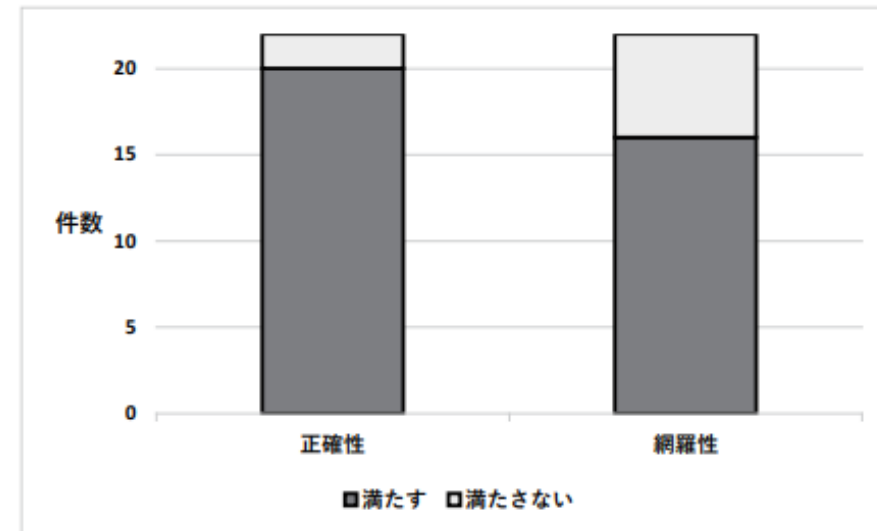
■ 使用技術抽出の評価指標

◆ 内容正確性

- 虚偽の内容が含まれていないか
- ファイルの内容を正しく説明しているかどうか

◆ 網羅性

- ニーズに関連する情報が出力に全て含まれているか



■ 考察

- ◆ 正確性は90%で、使用言語やライブラリ等を正確に抽出できた
 - 一般的な技術要素の抽出に役立つ
- ◆ 網羅性は73%で、満たさない例がいくつか存在
 - 研究室内独自のライブラリ・APIの使用などは抽出できなかった

UC2:組織内ソフトウェアに関連する実装の抽出



■ アップサイクルのコンテキスト

◆ 全体像がわかっていない

- 「SensorBoxManageService」では、どのようにSensorboxを管理する？

```
#instruction
```

あるソフトウェアのソースファイルを#input以下に示します。

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沿った情報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
```

ソフトウェア開発者は、以下のニーズを持っています。

```
needs:
```

このソフトウェアがどのようにSensorBoxを管理しているか知りたい

```
#input “”
```

```
{{input}}
```

```
#output
```



UC2の結果の一部

- ・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。

付加情報なし(UC2)の結果の一部



UC2の結果

■ 評価指標

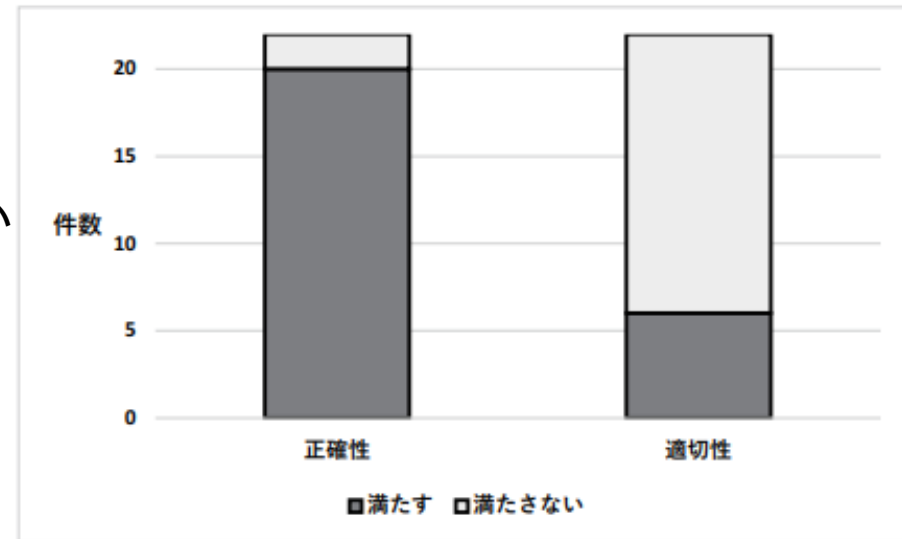
◆ 内容正確性

- 虚偽の内容が含まれていないか
- ファイルの内容を正しく説明しているかどうか

◆ 適切性

- ニーズと文脈に即した出力であるかどうか

例: 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→ 適切性を満たさないと判断



■ 正確性は90%で、多くが誤りのない情報であった

◆ 出力結果のほとんどが単なる実装の説明

■ 適切性は27%で、ニーズに則していない回答が多数

◆ 指示が抽象的であり、意図した回答が得られなかった

◆ 「ニーズに即した情報が存在しない」という回答ができなかった

A3: LLMを用いたソフトウェア全体からの素材抽出と評価



■ 開発が進み，出入力上限が急速に伸びてきている

◆ 例

- GPT-3.5 Turbo → 16,000トークン
- GPT-4 Turbo → **128,000**トークン

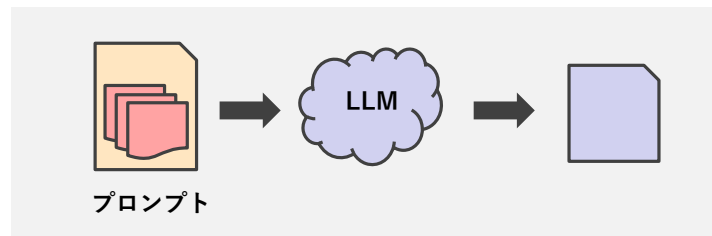
■ A2: 単一ソースファイルを対象としたLLM実行での限界

- ◆ ソースファイル間の連携やソフトウェア全体の構造情報は抽出できない

■ 目的

- ◆ LLMを用いて，ソフトウェア全体から素材抽出を行う

⇒ 1回のリクエストですべてのソースファイルを渡す





実験

■ 使用したモデル

- ◆ OpenAI API: GPT-4 Turbo

■ 実験対象: SensorBoxManageService

- ◆ LOC: 1565
- ◆ 総トークン数: 13,137

■ ユースケース

- ◆ Usecase1: アップサイクル素材の抽出指示を与える
 - 新規開発に役立つものを抜き出させる
- ◆ Usecase2: ソフトウェアのドキュメント情報とともに抽出指示を行う
 - 研究室wikiのSensorBoxに関する説明文書を使用



UC1で使したプロンプト

■ UC1:抽出指示のみを与える

```
#instruction
```

あるソフトウェアのソースファイルを#input以下に示します。

与えられた#contextに基づいて、それらのソースファイルをもとにソフトウェアの働きや構造を理解し、その働きを抽象化してください。

抽象化した情報から、全く別のソフトウェア開発に転用にできそうなアイデアや情報を示してください。

```
#context
```

必要に応じて、information:以下に示した情報を参考にしてください。

```
Information
```

```
なし
```

```
#input: ---
```

```
{{source_files}}
```

```
---
```

```
#output:
```



UC1: 結果の一部 (1/2)

提供されたソースファイルに基づくと、このソフトウェアはSensorBox Management Systemであり、温度、湿度、光、動き、音などの環境データを収集するために使用されるさまざまなセンサーボックスを構成および監視するために設計されています。このシステムは、おそらくスマートシティやスマートホームのアプリケーションに関連している可能性があり、大学や研究機関と提携しているかもしれません。

以下は、ソースファイルから抽象化された主要な機能です。

1. センサーの構成:

- ・XML構成ファイル (conf/sbox-phidget-*.xml) はさまざまなタイプのセンサーボックスを定義し、さまざまなセンサー(例: 温度、湿度、光)を持っています。
- ・センサーボックスは一意のIDで識別され、人が読める名前、所有者、場所、およびデータ通信のエンドポイントを持っています。
- ・ボックス内の各センサーは一意のID、デバイスタイプ、センサータイプ、アクセスメソッド(インターフェースタイプとポート番号を含む)、およびサンプリングレートを持っています。

2. エンドポイント管理 (endpoint.cgi):

センサーボックスのエンドポイントを管理するためのHTTPリクエストと対話するPerlスクリプト。ボックスの現在のエンドポイントを取得したり、新しいエンドポイントを設定したり、すべてのエンドポイントをリスト表示することができます。
出力用のXMLテンプレートを使用します (ep/endpoint.xml, ep/error.xml, ep/list.xml).

...



UC1: 結果の一部 (2/2)

他のソフトウェア開発プロジェクトでの再利用のためのアイデアと情報:

- ・センサーデバイスのためのモジュラーなXML構成は、デバイスの特性を定義する柔軟な方法であり、任意のIoTデバイス管理システムに適応できる可能性があります。
- ・エンドポイント管理システムは、センサーボックスだけでなく、エンドポイント構成を必要とするネットワークデバイスを管理するために一般化できます。
- ・ステータス監視スクリプトは、分散デバイスのヘルスチェックとステータス報告の一般的なアプローチを示しており、さまざまなコンテキスト(サーバーファーム、IoTネットワークなど)で適用できます。
- ・通知システムは、他の監視ソフトウェアでのアラートメカニズムを構築するためのテンプレートとして使用できます。
- ・ウェブインターフェースとその基になるJavaScriptコードは、バックエンドデータに基づいてUI要素を動的に更新するための出発点として、他のWebベースの管理コンソールの構築に役立つでしょう。



UC2で使用したプロンプト

■ UC2:ソフトウェアのドキュメント情報とともに抽出指示を行う

- ◆ LLMの学習データに存在しない情報(=SensorBoxに関するwikiページ)を付与

#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ソフトウェアの働きや構造を理解し、ソフトウェアの働きを抽象化してください。
抽象化した情報から、全く別のソフトウェア開発に

#context

必要に応じて、information:以下に示した

Information

{{related_information}}

#input: ---

{{source_files}}

#output:

センサボックスの登録に関する情報をまとめるページ

STEP1: センサボックス管理サービスへの登録

新しく作ったセンサボックスの定義ファイルを登録する

- TeratermがRLoginで、[redacted]にSSHログインする
 - [redacted]
- [redacted]になり、定義ファイルのディレクトリに移動. gitのブランチ server_configに

```
$ su
パスワード: [redacted]
# cd /var/[redacted]
```

- 登録済みの定義ファイル一覧を確認する

```
# ls
```

- 自分のセンサボックスの番号 sbox-phidget-XXXXXXX.confが存在しているか確認す



UC2: 結果

■ Wiki情報を踏まえた正しい出力が見られた

◆ UC1との差分

・センサーボックスのフレームワークの概念は、プラグアンドプレイのセンサー機能を備えたモジュラーなIoTシステムを作成するために適応できます。

→ プロンプトへの情報の補完による影響

◆ UC1の結果では見られなかった, SensorBoxの起動についての記述

- SensorBoxは起動時に設定ファイルを取得する

電源を繋いでから †

- センサボックスを電源に繋ぐと通信が始まる
 - 無線のランプが消えていたら電源を繋ぎなおす

センサボックス管理サービスの確認 †

- ブート時にセンサボックス管理サービスにIPアドレスを渡している
- <http://133.30.159.3/SensorBoxManageService/endpoint.cgi?cmd=list>を見ると
 - 最終更新日時
 - センサボックスのID
 - センサボックスのエンドポイントが表示される



まとめ・今後の展望

■ 目的

- ◆ 組織内における過去のソフトウェアプロジェクトからソフトウェアアップサイクルに利用可能な資源を抽出すること

■ 本研究における取り組み

- ◆ A1: メソッド名構成単語に基づく概要推測に関する検討と評価
 - tagcloud画像を用いることで機能に関する特徴の表現
- ◆ A2: LLMを用いた単一ソースファイルからの素材抽出と評価
 - ソースファイルに含まれる技術要素を90%の正確性で抽出
- ◆ A3: LLMを用いたソフトウェア全体からの素材抽出と評価
 - 複数のソースファイルから構成される特徴をLLMによって抽出
 - ドキュメント情報の活用により特定ドメインに関する説明精度が向上

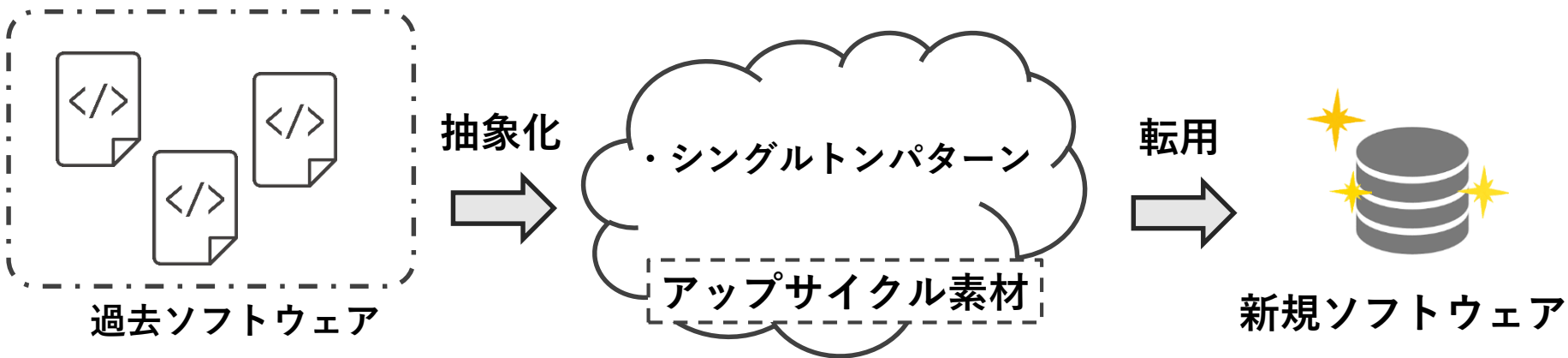
■ 今後の展望

- ◆ 抽出素材の分類
- ◆ LLMを活用した抽出素材の組み合わせ



背景: ソフトウェアアップサイクル

- ソフトウェアプロジェクトが持つ具体的な構造や設計思想, 理念などを抽象化し, 別のソフトウェア開発に転用すること
 - ◆ 組織において放置されがちな既存ソフトウェアに着目
 - ◆ 既存資源の再利用(=アップサイクル)による開発運用の効率化を目指す
- 例1: デザインパターン
 - ◆ ソフトウェア設計における一般的な問題に対する解決策
 - ◆ 誰でも利用可能なレベルまで抽象化(一般化)されたアイデア





背景: 大規模言語モデルの発展

■ 大規模言語モデル

- ◆ 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
 - 入力文をトークンと呼ばれる単位に変換し、生成を行う
- ◆ ソフトウェアの分野でも利用が広がる
 - ChatGPT、Github Copilotなど

■ LLMの限界

- ◆ 学習データが含まれない情報の回答に弱い
 - 組織の非公開情報やドメイン知識についての回答生成は困難
- ◆ 対応策
 - Fine-tuning
既存の訓練済みモデルから派生させ、特定のタスクに特化
 - In-context Learning
コンテキスト長の制限内で、プロンプト内に補足情報を含める



実験設定

■ 対象: SensorBoxManageService

- ◆ SensorBox (IoTデバイス) を管理するWebサービス
- ◆ LOC: 1565
- ◆ 総トークン数: 13,137
- ◆ 使用技術: SensorBox, PHP, Perl, javascript, css



SensorBox

SensorBox Management Console

[Console for Type2 SensorBox](#)

DeviceID	Name	Location	IP address	Time Series	Current State	Last Updated on	Maintainer
sbox-phidget-148989	百葉箱センサボックス	外の百葉箱	192.168.2.37	View	View	2024-02-11T12:54:42 (ONLINE)	インフラ班
sbox-phidget-458618	中村実家センサボックス1	リビング	192.168.1.9	View	View	2024-02-11T12:54:42 (ONLINE)	中村 忠雄
sbox-phidget-458814	中山家センサボックス	テレビの横	192.168.40.2	View	View	2024-02-11T12:54:40 (ONLINE)	中山邦子
sbox-phidget-458906	中村実家センサボックス3	設置待ち	192.168.1.185	View	View	2024-02-11T12:54:40 (ONLINE)	中村 忠雄

SensorBoxManageServiceの管理画面



プロンプトエンジニアリング

■ Zero-Shot

- ◆ 具体例を示さず，指示のみを記述

■ Few-Shot

- ◆ 出力の具体例を示しつつ指示を記述

■ Chain-of-Thought

- ◆ 中間的な推論ステップを用いる

このグループの奇数を合計すると偶数になります。: 4、8、9、15、12、2、1。

A: 奇数を全て加えると(9, 15, 1)25になります。答えはFalseです。

このグループの奇数を合計すると偶数になります。: 15、32、5、13、82、7、1。

A:

■ Zero-shot COT

私は市場に行って10個のリンゴを買いました。隣人に2つ、修理工に2つ渡しました。それから5つのリンゴを買って1つ食べました。残りは何個ですか？

ステップバイステップで考えてみましょう。



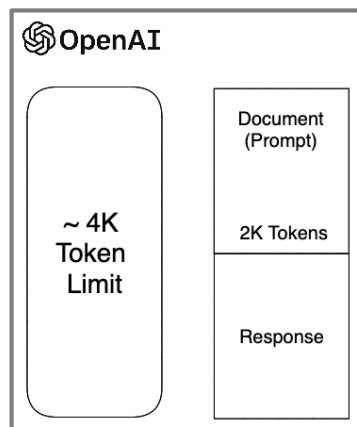
コンテキスト長制限下における工夫

■ Stuff

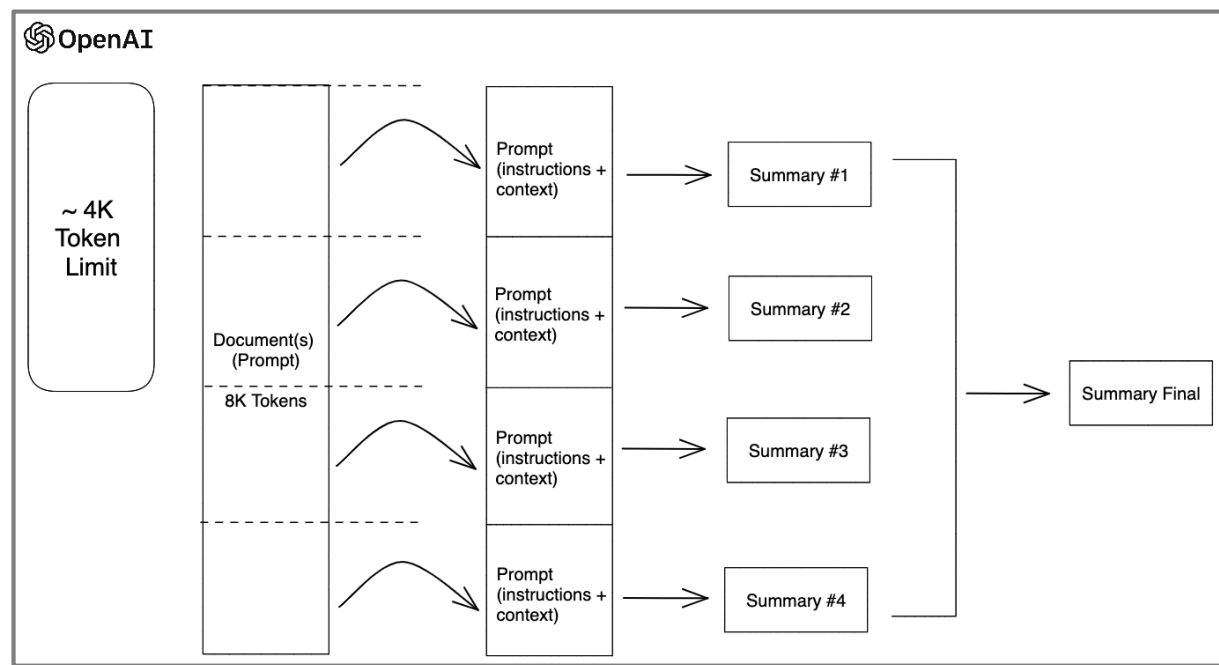
- ◆ 一つの入力ですべてのテキスト処理を行う

■ Map-Reduce

- ◆ 入力を分割し、最終的にそれぞれの出力結果を統合する



Stuff



Map-Reduce

引用: <https://brain.d.foundation/Engineering/AI/Workaround+with+OpenAI's+token+limit+with+Langchain>



実装

- プログラミング言語: Python
- リポジトリマイニング対象: 中村研究室Gitlab
- モデル: OpenAI GPT-3.5 Turbo / GPT-4 Turbo
- ライブラリ: LangChain
 - ◆ プロンプトテンプレート機能の利用



成果2: 実験

■ Usecase1: 使用技術の抽出

- ◆ 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

■ Usecase2: 組織内ソフトウェアに関連する実装の抽出

- ◆ 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

■ Usecase3: 補足情報を含めた組織内ソフトウェアに関連する実装の抽出

- ◆ related_information:

- 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



成果3: Usecase2で用いたソフトウェアドキュメント

- 付加情報として, SensorBoxに関する説明文書を利用
 - ◆ SensorBoxの概念や具体的な使い方に関する説明

センサボックスの登録に関する情報をまとめるページ

STEP1: センサボックス管理サービスへの登録[†]

新しく作ったセンサボックスの定義ファイルを登録する

- TeratermかRLoginで, [redacted] にSSHログインする
 - [redacted]
- [redacted] になり, 定義ファイルのディレクトリに移動. git のブランチ server_config に

```
$ su
パスワード: [redacted]
# cd /var/lib/ssh/sensor-box-manager-1100/ssh
```

- 登録済みの定義ファイル一覧を確認する

```
# ls
```

- 自分のセンサボックスの番号 sbox-phidget-XXXXXXX.confが存在しているか確認す



他のソフトウェアへの適用

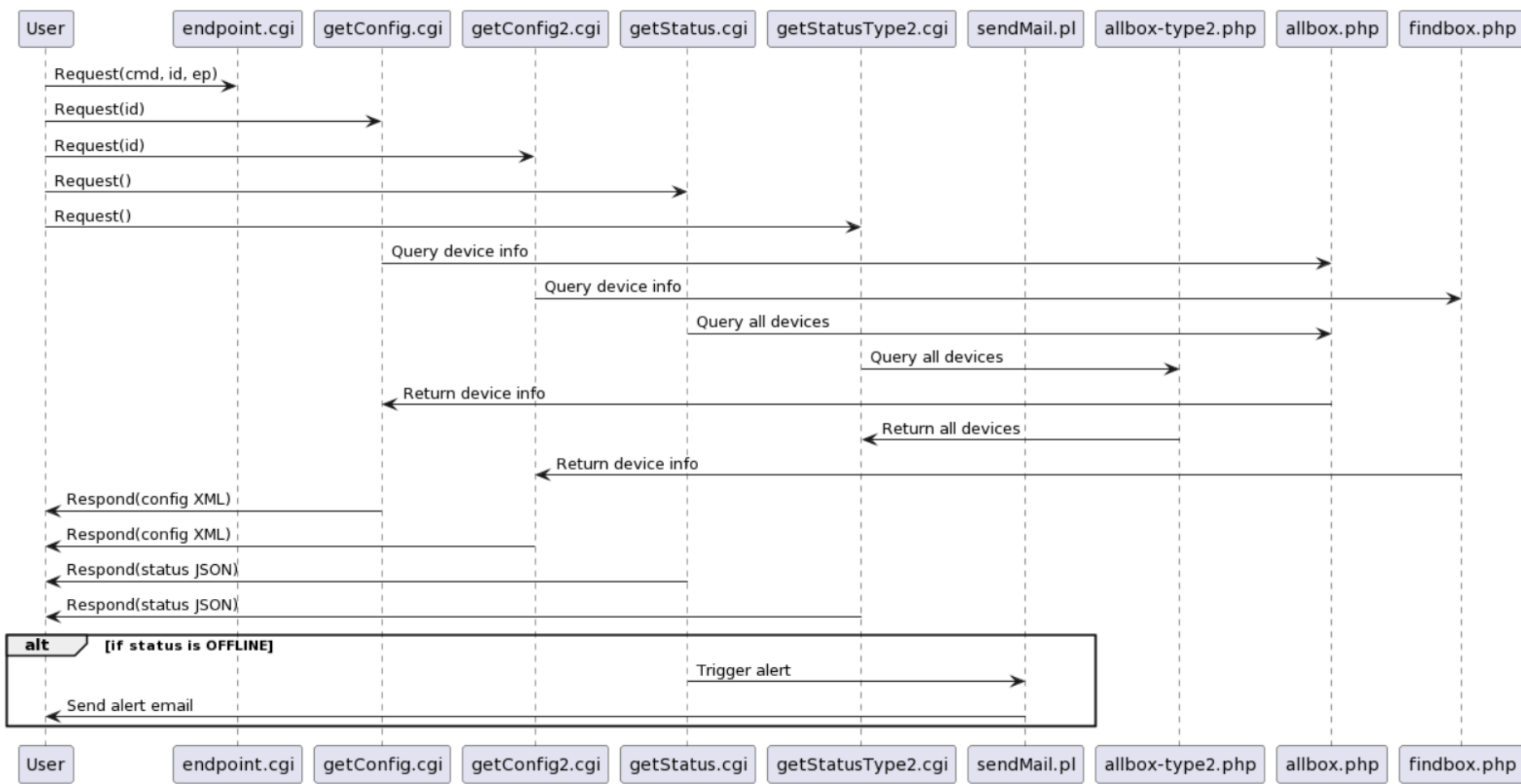
■ 消費電力管理ツール

◆ IoT機器を用いて消費電力を計測・可視化するサービス

- **API通信レイヤー:** ``src/api/`` ディレクトリの構造は、異なるAPIエンドポイント用のモジュールな関数を持つ他のプロジェクトに適応できます。
- **データ可視化コンポーネント:** ``GraphCard`` コンポーネントは、時系列データやその他のメトリクスを視覚化するために他のプロジェクトで再利用できます。
- **開発および本番用のDocker構成:** ``docker-compose`` や ``Dockerfile`` を使用したDockerのセットアップは、他のWebアプリケーションのビルドとデプロイメントのテンプレートとして機能します。
- **TailwindCSSを使用したスタイリング:** TailwindCSSを使用したスタイリングは、迅速なUI開発のために他のプロジェクトで採用できます。



LLMによるシーケンス図(PlantUML)





A3: 結果と考察

■ 結果

- ◆ Usecase1, 2について、抽象的な設計情報を抽出できた
 - 「xmlファイルによる設定管理」という本質に関する記述

このソフトウェアから他のソフトウェア開発プロジェクトに再利用できる可能性のあるアイデアと情報には次のようなものがあります:

・モジュラーなXML構成は、デバイスの特性を柔軟に定義できるため、異なる種類のIoTデバイス管理システムで採用できます。新しいデバイスの追加や既存デバイスの変更が容易です。

Usecase1の結果の一部

■ 考察

- ◆ LLMの利用により、様々な抽象度の素材を抽出できることを確認した
 - 技術要素, 設計情報など

- ◆ Usecase2では、既存情報を付与することで正確性が向上した

⇒ソフトウェアにまつわるドキュメントをプロンプトに用いることで
抽出内容の精度向上が期待できる