

技術的負債の 詳細文生成手法の構築

オープンソースソフトウェア工学研究室

学籍番号：S2420102

氏名：野口 優輝

技術的負債 (Technical Debt)

ソフトウェア開発において、納期やコストの制限によって導入される質の低い設計やソースコード^[1]

例) スパゲッティコード：構造が複雑で理解が難しいソースコード
コードスメル：深刻な問題を抱えているソースコードの特徴

技術的負債が蓄積していくと . . .

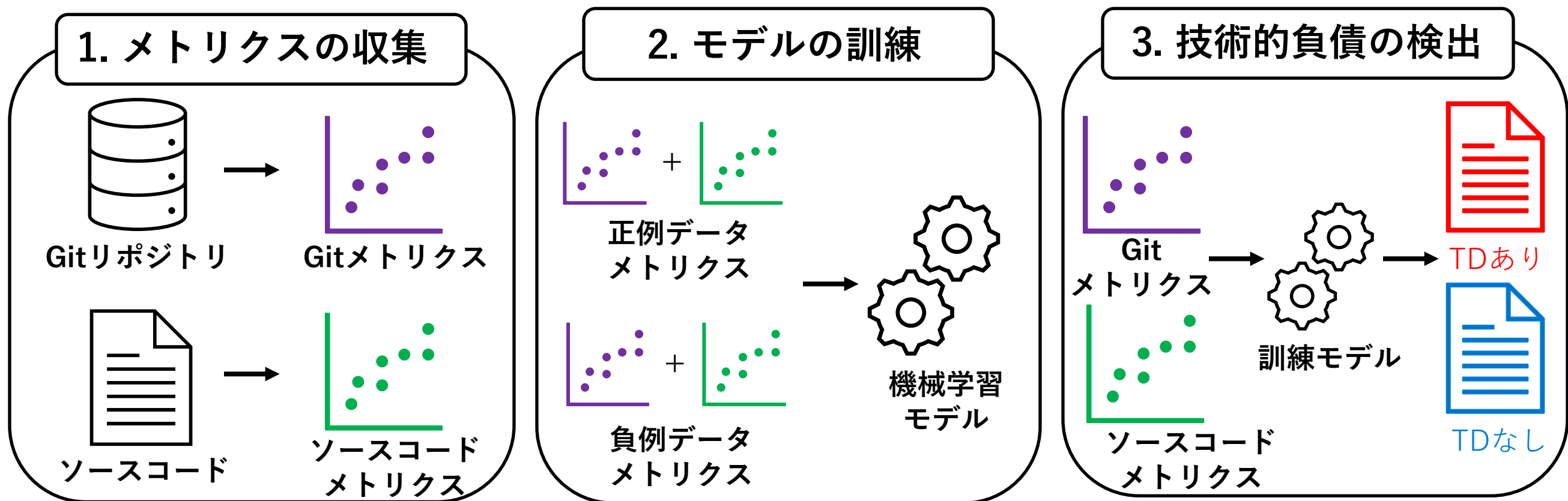
- ソフトウェアの保守性の低下, 機能欠陥を引き起こす
→ 除去するための保守コストが増大

技術的負債を早期に検出し除去されることが望まれる

[1]Ward Cunningham. The wycash portfolio management system. SIGPLAN OOPS Mess., Vol. 4, No. 2, pp. 29–30, 1992.

技術的負債を検出する研究

機械学習を使って、ソースコード中のクラスに含まれている技術的負債を検出^[2]

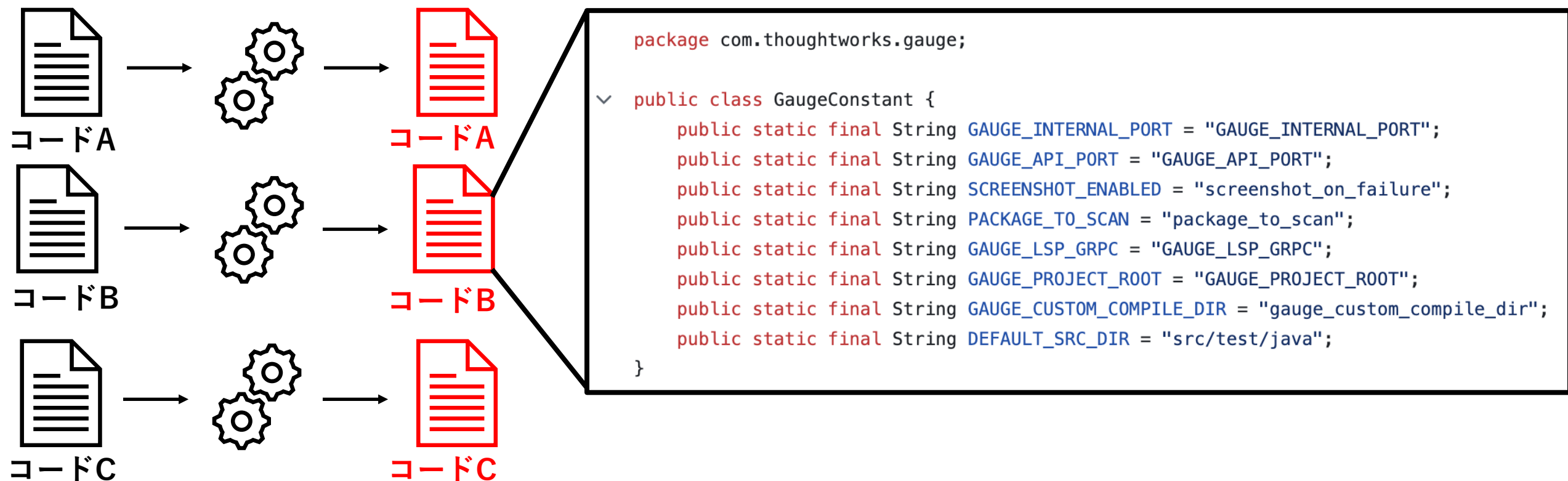


より高い精度で技術的負債を検出でき、技術的負債の管理が促進

[2] D. Tsoukalas, A. Chatzigeorgiou, A. Ampatzoglou, N. Mittas and D. Kehagias, "TD Classifier: Automatic Identification of Java Classes with High Technical Debt," 2022 IEEE/ACM International Conference on Technical Debt (TechDebt), Pittsburgh, PA, USA, 2022, pp. 76-80

技術的負債の検出研究：課題点

技術的負債を検出するだけで、具体的な詳細がわからない



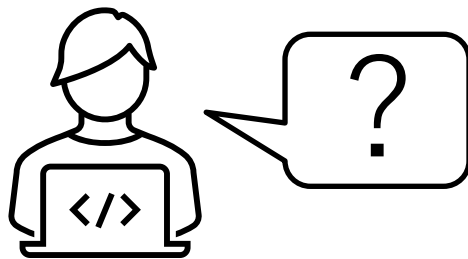
- 技術的負債を検出できても、開発者が理解できないといった状況が起こってしまう

→技術的負債の管理などに無駄なコストがかかってしまう可能性

技術的負債の検出研究：課題点

技術的負債を検出するだけで、具体的な詳細がわからない

負債がなぜ発生したのか
どの負債を修正すべきなのか
どの程度深刻な負債なのか



```
package com.thoughtworks.gauge;  
  
✓ public class GaugeConstant {  
    public static final String GAUGE_INTERNAL_PORT = "GAUGE_INTERNAL_PORT";  
    public static final String GAUGE_API_PORT = "GAUGE_API_PORT";  
    public static final String SCREENSHOT_ENABLED = "screenshot_on_failure";  
    public static final String PACKAGE_TO_SCAN = "package_to_scan";  
    public static final String GAUGE_LSP_GRPC = "GAUGE_LSP_GRPC";  
    public static final String GAUGE_PROJECT_ROOT = "GAUGE_PROJECT_ROOT";  
    public static final String GAUGE_CUSTOM_COMPILE_DIR = "gauge_custom_compile_dir";  
    public static final String DEFAULT_SRC_DIR = "src/test/java";  
}
```

プログラム実行時に、GAUGE_API_PORTが今後不要となる

- 技術的負債を検出できても、開発者が理解できないといった状況が起こってしまう

→技術的負債の管理などに無駄なコストがかかってしまう可能性

TD Issue

課題票（Issue）にTDラベルを割り当てて明示される

Issue

課題管理システムに記録される報告のこと

不具合や機能欠陥など様々な課題が管理されている

タイトル

techdebt: use of APNs binary format #26

New issue

本文

AndrewX192 opened on Apr 25, 2019

While setting up push notifications for the iOS application, I noticed that pushy (the library, not the service) is using a version from 2015 which pre-dates Apple's transition to HTTP/2. I'm not sure how much longer this will be supported but it seems like the new HTTP/2 format is easier to work with and the path forward. We should upgrade the pushy library and switch to HTTP/2.

Assignees

No one assigned

Labels

techdebt

Type

No type

techdebt

‘debt’などのTDに関するラベルが貼られる

AndrewX192 added techdebt on Oct 27, 2019

TD Issueの例

- TD Issueの本文には技術的負債について記載される
 - 技術的負債の問題点などを理解することができる

TD Issue：問題点

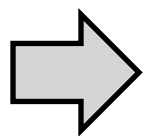
TD Issue本文での質が悪い

- 技術的負債を理解するために
 - どのような症状を持っているか
 - どのような影響をもたらすのか
 - 技術的負債を防ぐには，将来的にどのような設計にするべきか
- 実際のTD Issue本文

TD Issue本文例

例 1) **Add a test to NetworkIsServiceTest** witch cover changes from #2207

例 2) We will **have a file** that gets all the possible elements for well and wellbore and then will **pass that through the merge logic.**



将来行うべき内容のみの記載で，実際に技術的負債が
どんな症状・影響を与えているのかわからない

目的・アプローチ

目的

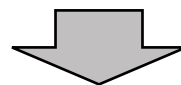
- 検出された技術的負債の詳細文（TD Issue本文）を自動生成する手法の構築

アプローチ

- 生成AIを用いて詳細文を生成

----- 生成AIを用いる利点 -----

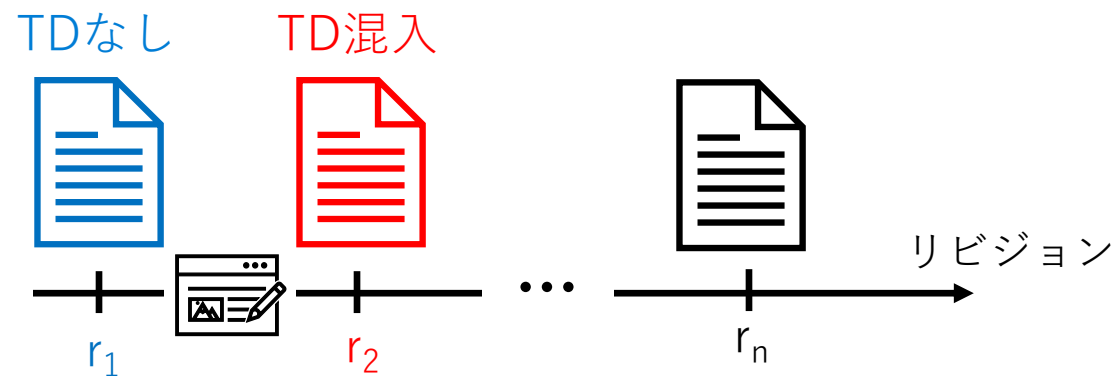
- ・ファインチューニングを必要としない
- ・柔軟な文章を生成することができる



学習にかかるコストを削減でき、
プロンプトの最適化から目的に応じた詳細文を生成できる

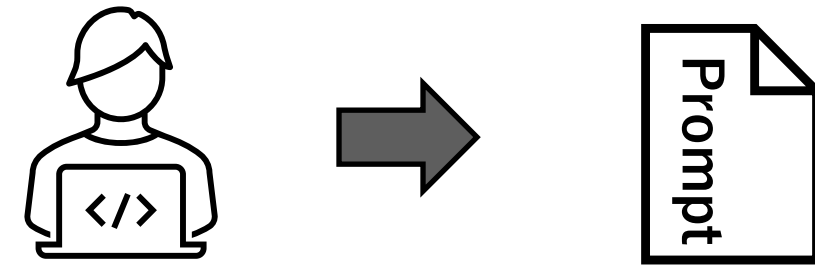
アプローチ：概要

1. TD混入前後のファイルを特定



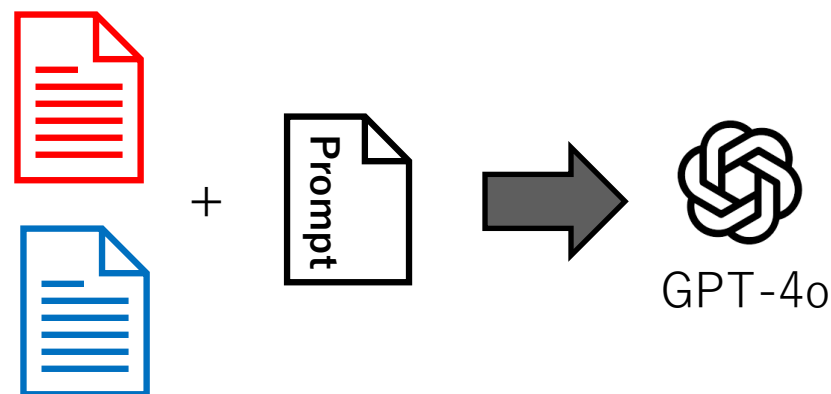
TDコード片が初めて出現した
リビジョンからTD混入コミット
を特定し、前後のファイルを取得

2. プロンプトの作成



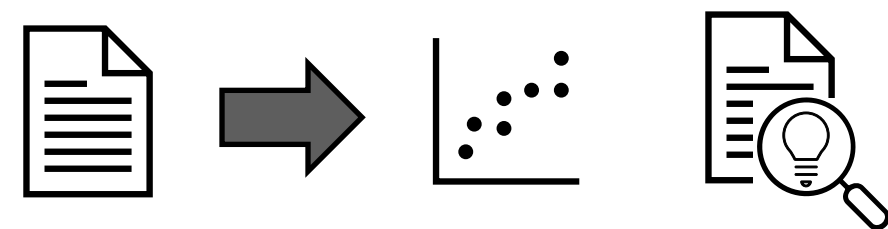
TDが混入する前後の振る舞いについて
説明するようにプロンプトを作成

3. 生成AIにプロンプトを入力



二つのコードをプロンプトに
組み込み詳細文を生成

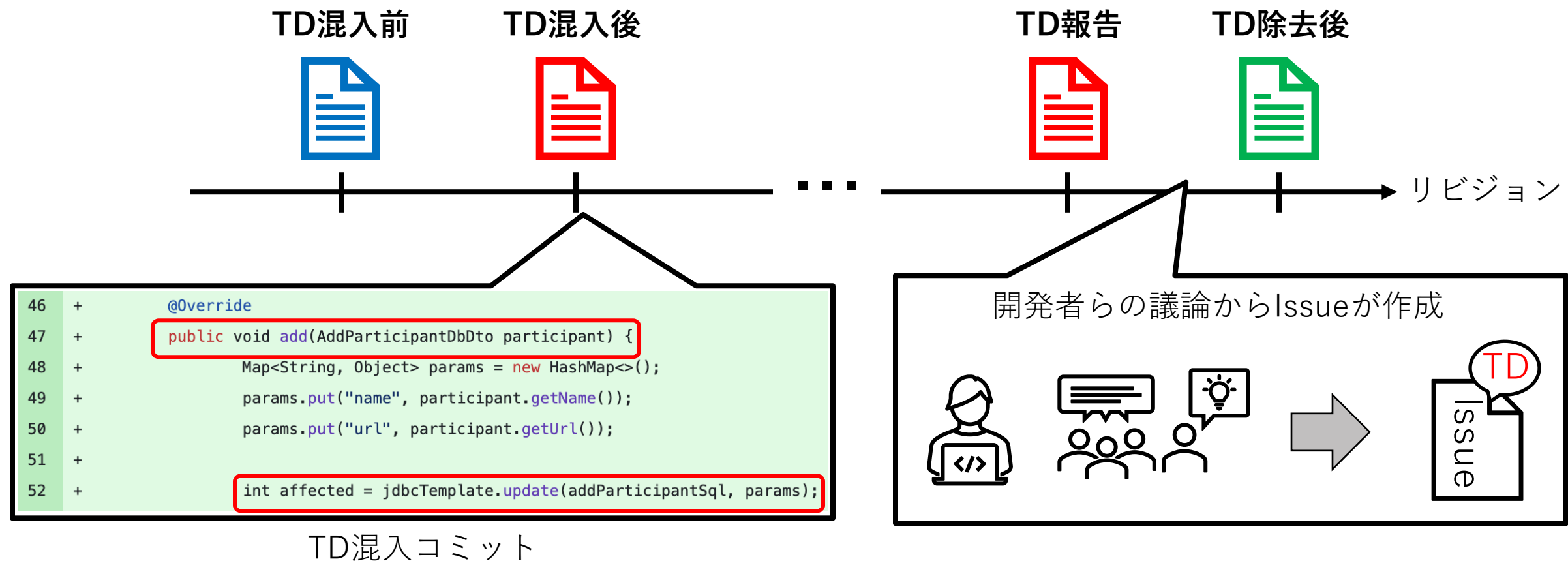
4. 生成された詳細文の評価・分析



- ・ 意味的類似度で評価
- ・ 結果を目視し、
人手で詳細文を評価する

TD混入前後のファイル特定

TDが混入してから除去までの流れ



- TDが混入することで、動作の振る舞いに違いが生じる
 - この振る舞いの違いについてがIssueとして報告される
- TD混入前後のファイル情報を使って、詳細文（Issue本文）を生成

プロンプト作成

You are an AI assistant tasked with analyzing the behavioral differences between two versions of a code and reporting the issues you find as a GitHub Issue.

技術的負債の分析の役割説明

Your task is to compare the behavior of the following two code snippets:

1. Code without technical debt
2. Code with technical debt

分析対象のソースコード説明

技術的負債あり・なしの
2つの動作を比較することを指示

Technical debt refers to poor quality design or source code introduced due to short-term deadlines or cost constraints. The code you will compare represents the code before and after a change introduced technical debt.

技術的負債の定義・分析目的の説明

Based on the behavioral differences in the code with technical debt, generate an issue body that explains the impact of introducing technical debt on the system's behavior and what future design should be adopted to resolve this impact.

- ・システムの動作に与える影響
- ・影響を解消するための今後の設計方針

Here is code without technical debt and code with technical debt

Code without technical debt : {技術的負債混入前コード}

Code with technical debt : {技術的負債混入後コード}

比較するソースコードの提示

Output Format

Detail the differences in behavior :

Explain the impact :

Propose a future design:

分析結果の出力フォーマット

- ・動作の違い
- ・影響を説明
- ・将来の設計の提案

データセット

既存研究^[3]のデータセットをフィルタリングし利用

- Javaプロジェクトのリポジトリに含まれているIssueを対象

Javaプロジェクトを使用する理由

多くの大規模プロジェクトがあり、技術的負債の研究が盛んに行われている

- **フィルタリング**
 - 公開されているリポジトリ
 - 過去に技術的負債の除去が行われている
 - Issue本文が空白でない
 - Issue本文がBotで作成されていない

プロジェクト数	TD Issue
140	831件

[3] D. Skryseth, K. Shivashankar, I. Pilán and A. Martini, "Technical Debt Classification in Issue Trackers using Natural Language Processing based on Transformers," In Proceedings of 6th International Conference on Technical Debt (TechDebt), pp. 92-101, 2023

実際にプロンプトから生成

/**Detail the differences in behavior:**

メソッドlookupInCustomStoreに動作の違いがある
技術的負債のあるコードでは～
技術的負債のないコードでは～

Explain the impact:

1. 複雑さと依存関係の増加
2. 柔軟性の低下
3. エラーのリスク

Propose a future design:

1. key読み取りロジックの分離
2. 依存性注入の利用
3. 後方互換性の維持

```
public static JKey lookupInCustomStore(String storeLoc, String keyName) {  
    try {  
        Map<String, List<AccountKeyListObj>> store = readBase64EncodedGenesisKey(storeLoc);  
        List<AccountKeyListObj> objs = store.get(keyName);  
        var pubKey = objs.get(0).getKeyPairList().get(0).getPublicKeyAbyteStr();  
        return new JEd25519Key(commonshexToBytes(pubKey));  
    } catch (InvalidKeySpecException e) {  
        throw new IllegalArgumentException(e);  
    } catch (DecoderException e) {  
        throw new IllegalArgumentException(e);  
    }  
}
```

技術的負債混入前コード



```
public static JKey lookupInCustomStore(LegacyEd25519KeyReader b64Reader, String storeLoc, String kpId) {  
    try {  
        return new JEd25519Key(commonshexToBytes(b64Reader.hexedABytesFrom(storeLoc, kpId)));  
    } catch (DecoderException e) {  
        throw new IllegalArgumentException(e);  
    }  
}
```

技術的負債混入後コード

- 二つのコード間での振る舞いの違いから発生する技術的負債を説明できてそう？
 - 発生する影響やそれに対する将来の設計の妥当性は？

今後の予定

作成したプロンプトから詳細文を生成・分析

- 大体API料金は40ドルくらい？

詳細文の評価

- 手動評価になる？
 - 大前提, Issueの内容が悪いということから, Issue本文を正解にできない
 - 評価項目を決定し, 2人それぞれの kappa 係数から詳細文を評価

TD解決後のファイル情報も加える

- TD混入前後・解決後のファイルから詳細文を生成
 - TD解決後でも理想とした振る舞いを示すかどうか