

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
S2420102 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 . . .

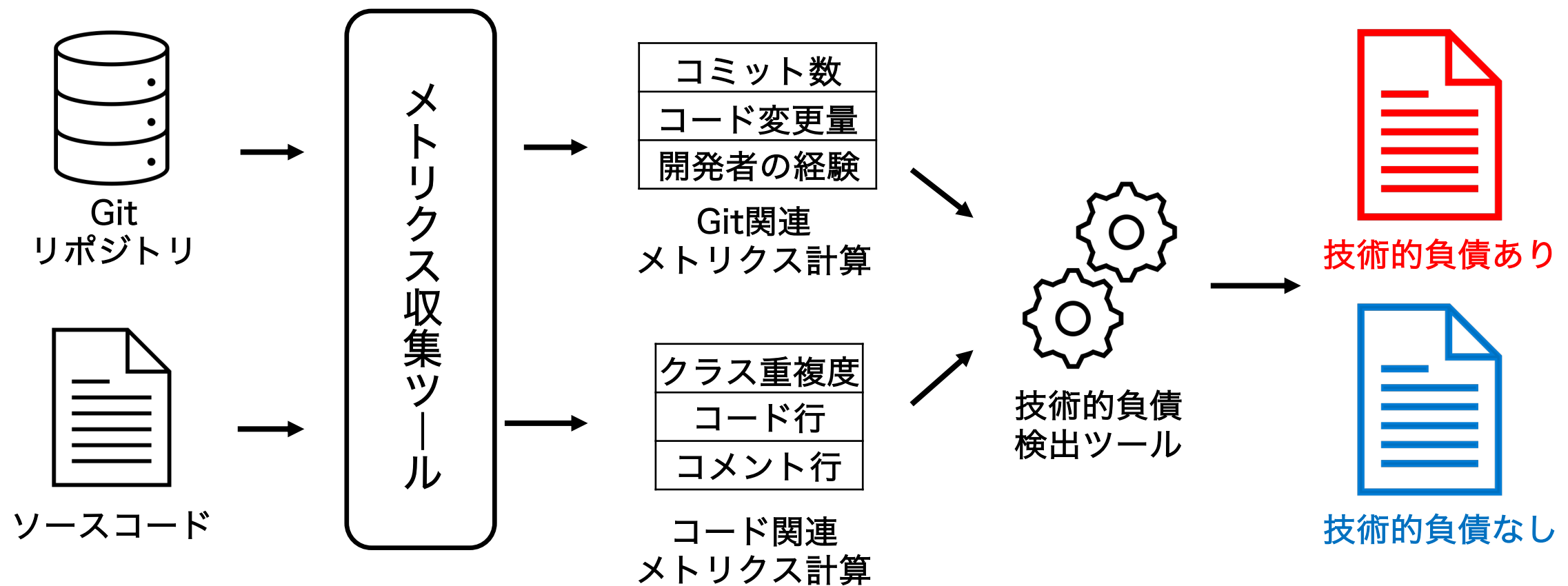
- ソフトウェアの保守性の低下, 機能欠陥を引き起こす
→ 除去するための保守コストが増大

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]

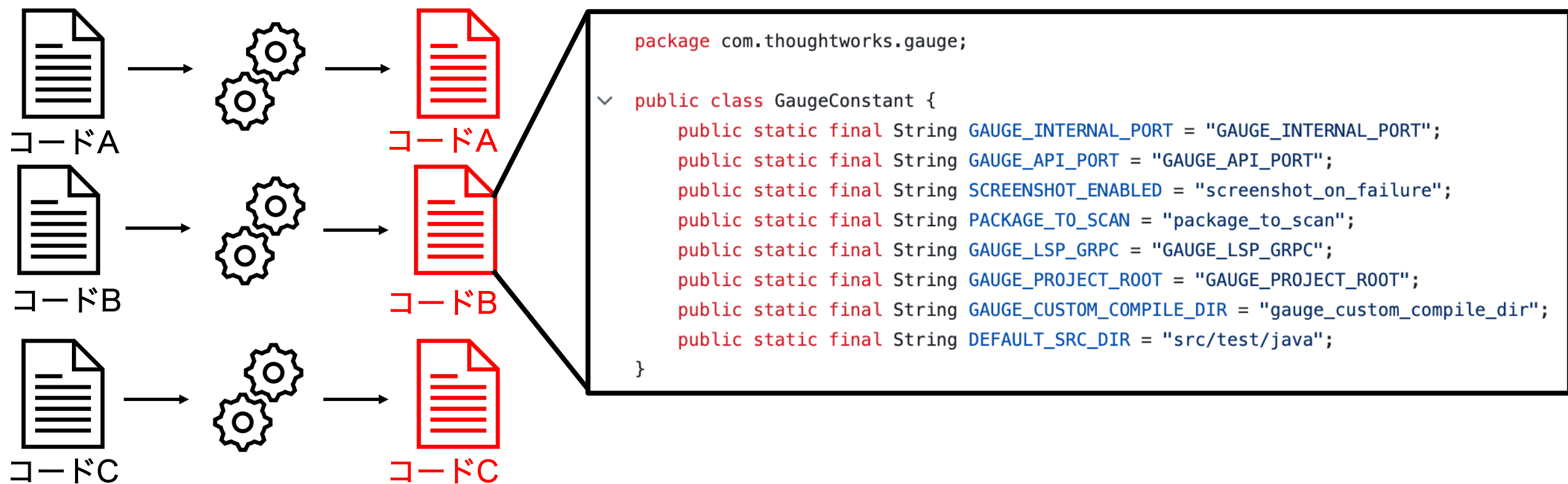


より高い精度で技術的負債を検出でき、技術的負債の管理が促進

[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

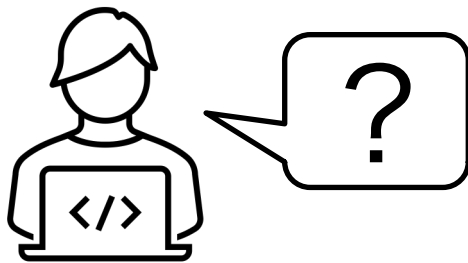


- 技術的負債を検出できても、開発者が理解できないといった状況が起こってしまう
→ 技術的負債の管理などに無駄なコストがかかってしまう可能性

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

目的とアプローチ

目的

- 検出された技術的負債の詳細文を自動生成する手法の構築

アプローチ

- Issueを使って事前学習済みモデルをファインチューニングし、
詳細文（タイトル・本文）を生成

-- Issue

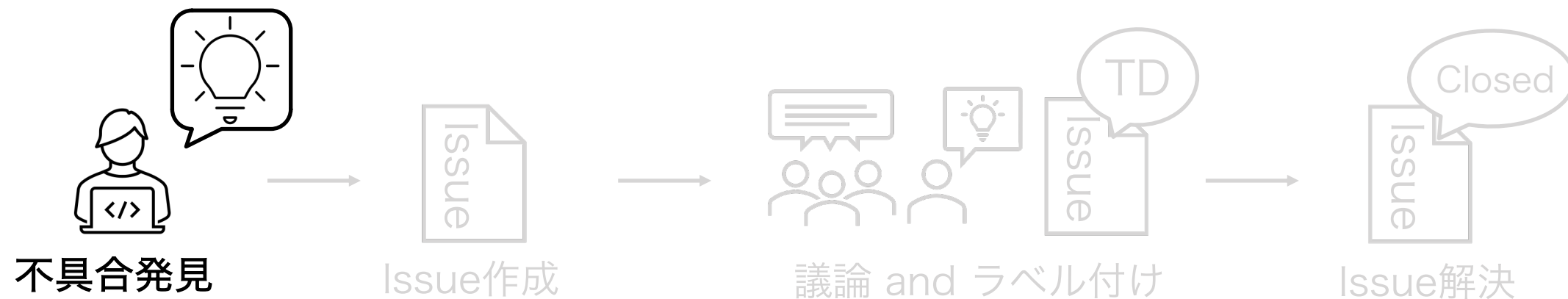
課題管理システムに記録される報告のこと

不具合や機能欠陥など様々な課題が管理されている

ラベルが付与され、**"debt"**が付与されると技術的負債に関する報告になる

技術的負債が解決されるまで

技術的負債を含むコードがツールを使って検出される



```
package com.thoughtworks.gauge;
```

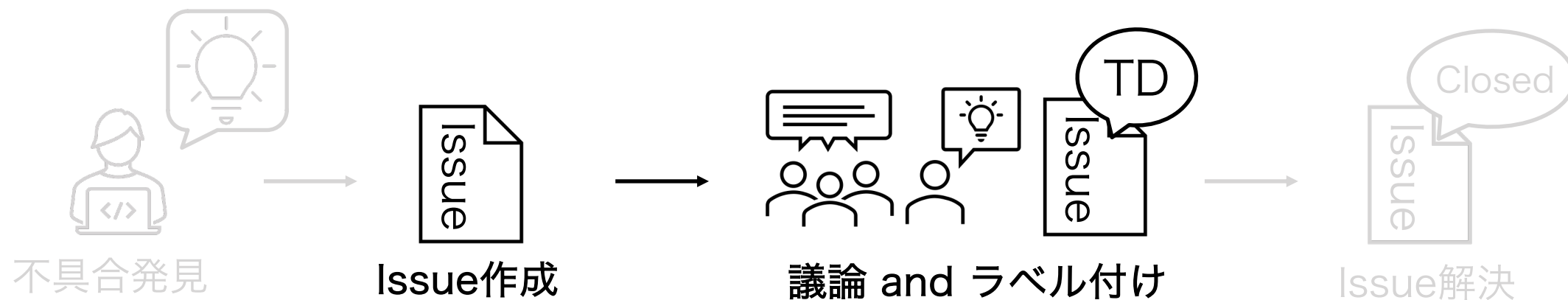
```
public class GaugeConstant {  
    public static final String GAUGE_INTERNAL_PORT = "GAUGE_INTERNAL_PORT";  
    public static final String GAUGE_API_PORT = "GAUGE_API_PORT";  
    public static final String SCREENSHOT_ENABLED = "screenshot_on_failure";  
    public static final String PACKAGE_TO_SCAN = "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";  
}
```

検出されたソースコード

検出されたソースコード中の
どの部分に技術的負債が
含まれているのかわからない

技術的負債が解決されるまで

開発者らの議論によってラベルが付与される



Remove the use of GAUGE_API_PORT #250

✓ Closed nehashri opened this issue on Feb 20, 2019 · 1 comment



nehashri commented on Feb 20, 2019

Contributor ...

We need not use `GAUGE_API_PORT` for fetching step value anymore during execution, this logic already exists Java plugin. The connection to this port can be removed.



Assignees

Apoorva-GA

Labels

tech debt technical

Labels

tech debt

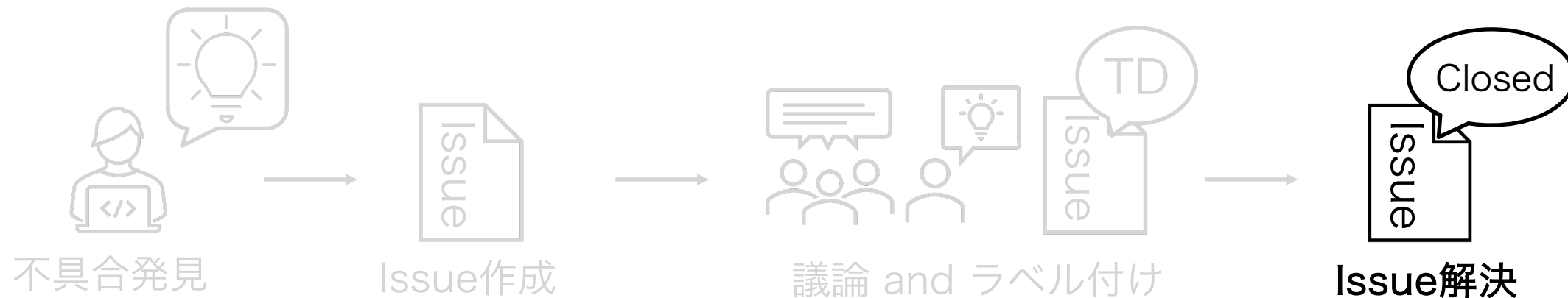
technical

→ TD Issueと呼ぶ

開発者らによって
技術的負債の議論がされる

技術的負債が解決されるまで

技術的負債を除去し，Issueが解決される



```
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";  
}
```

TDの除去コミット

開発者らの議論をもとに，
技術的負債を除去

TD Issueの概要

タイトル

techdebt: use of APNs binary format #26

🔒 Closed AndrewX192 opened this issue on Apr 25, 2019 · 2 comments · Fixed by #215

技術的負債を簡潔に要約

本文

AndrewX192 commented on Apr 25, 2019

Collaborator ...

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.



🏷️ AndrewX192 added the **techdebt** label on Oct 27, 2019

技術的負債の説明

発生している状況や環境など

議論

AndrewX192 commented on Nov 6, 2019

Collaborator Author ...

The version of pushy used now holds notification-services back on Netty 4.0 and generally means we can't upgrade netty for all of arcusplatform.



AndrewX192 commented on Nov 6, 2019

Collaborator Author ...

This must be completed by Nov 2020: <https://developer.apple.com/news/?id=11042019a>



🔗 AndrewX192 mentioned this issue on Apr 30, 2020

Update Pushy to support HTTP2 #215

🔗 Merged

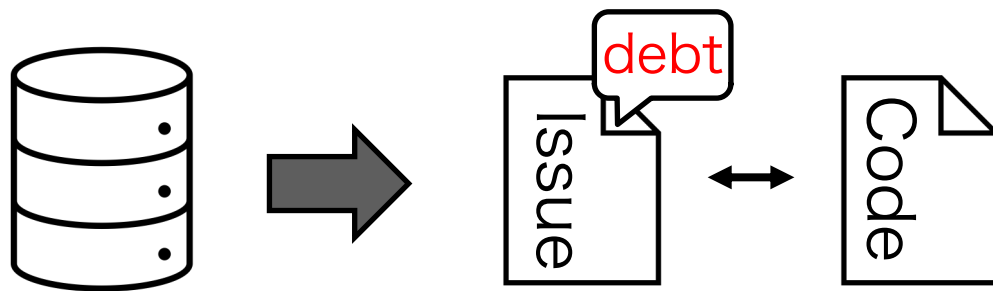
🔒 AndrewX192 closed this as completed in #215 on May 1, 2020

技術的負債をどのようにして
解消していくか

Closed

アプローチの概要

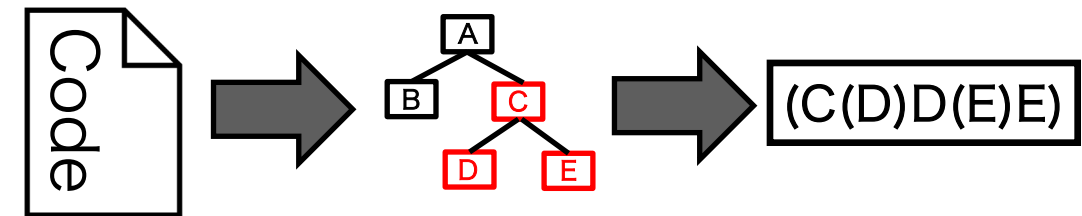
1. データ収集



リポジトリから以下のデータを収集

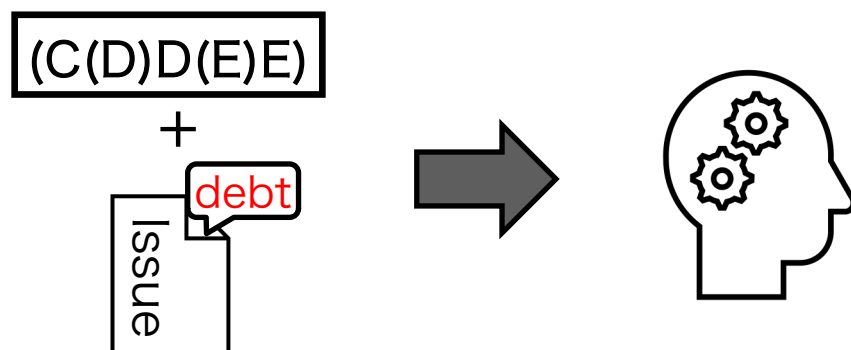
- ・ TDラベルが付与されたIssue
- ・ Issueに紐づいたソースコード

2. SBTシーケンスに変換



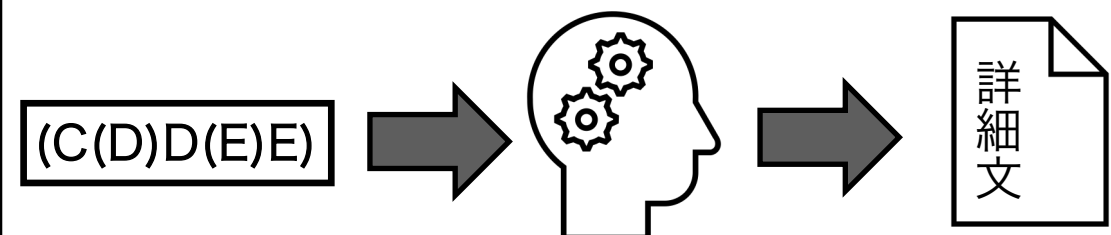
- ・ ソースコードをASTに変換
- ・ ASTからSBTシーケンスに変換

3. モデルをファインチューニング



- ・ 使用モデル：CodeT5
- ・ データをベクトル化しモデルに入力

4. 詳細文を生成・評価



- ・ SBTシーケンスから詳細文生成
- ・ 詳細文とIssueの意味的類似性から評価

SBT (Structured-Based-Traversal) [3]

ソースコードの構造情報が理解できる

- ASTからソースコードの構造情報（構文的, 依存関係）を保持したままシーケンスに変換
→SBTシーケンス

元のツリー構造：



SBT シーケンス：

(A(B)B(C(D)D(E)E)C)A

(A：Aノードの始まり

(B)B：BノードはAの子であり, Bには子ノードがない

(C(D)D(E)E)C：Cノードの中にDとEがある

)A：Aノードの終了

データセット

Javaプロジェクトのリポジトリを対象

- Java：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
- 既存研究^[4]のデータセットをフィルタリングして使用

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本文が空白なもの
- Issue本文がBotで作成されたもの

プロジェクト数	TD Issue
140	1,021件

データセット

[4] 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

データセット前処理

TD Issue

- データのクリーニング^[4]
 - メールアドレス, URL, 内部参照, 署名, 名前, マークダウン見出し
→削除
 - version文字列→'version', commit_hash→'sha', 数字→'0'に置換

ソースコード

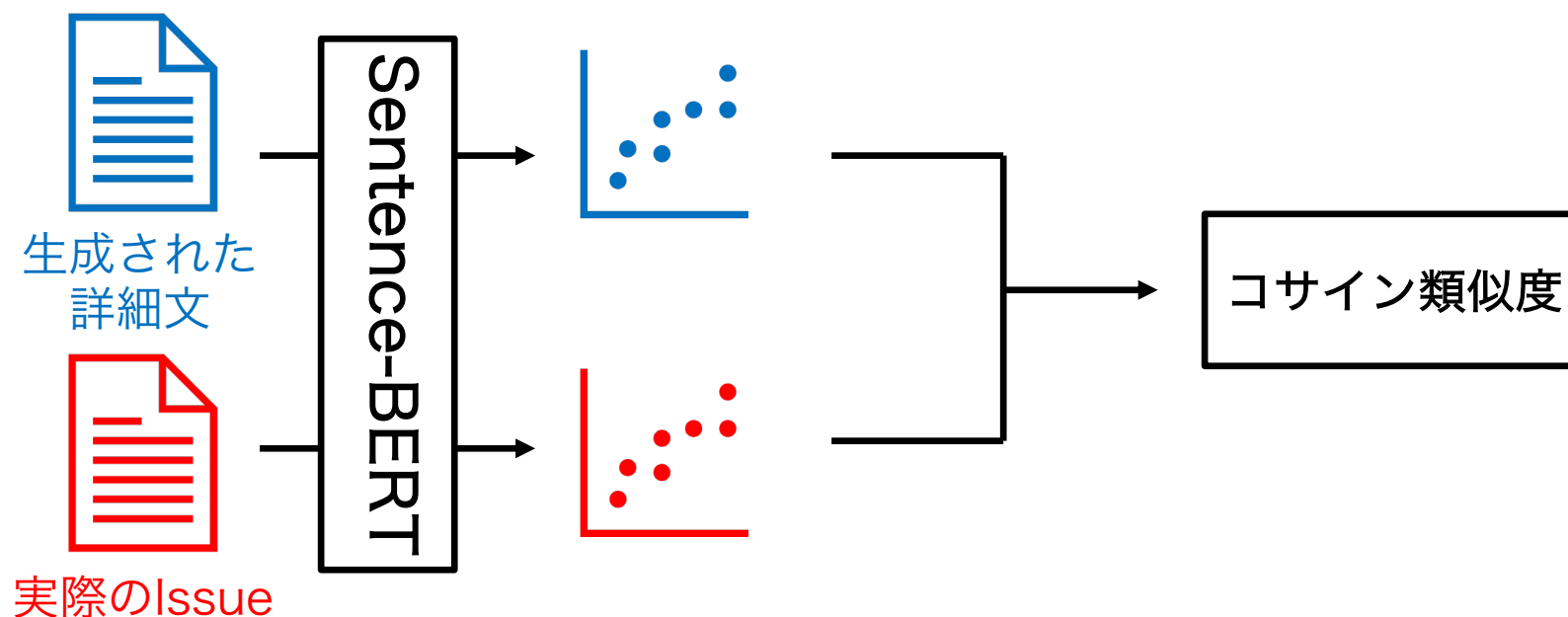
- TDを含むソースコード片をSBTシーケンスに変換

[4]Zhongxin Liu, Xin Xia, Christoph Treude, David Lo, and Shanping Li. 2019. Automatic generation of pull request descriptions. In 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). 176–188.

評価方法

生成された詳細文とIssueの意味的類似性で評価

- 生成された詳細文と実際のIssueをそれぞれベクトル化
- cos類似度を計測し，二つの文章の意味的類似度を計測



Sentence-BERT[?]

- 文全体の意味的類似度を効率的に計算し，最適化する手法

実験設定

パラメーター

- エポック数：5
- バッチ数：16
- オプティマイザー：Adam
- 損失関数：Cross Entropy

データセット分割

- 訓練：検証：テスト = 8：1：1 に分割して実施
 - Issueデータ→訓練：767件， 検証：127件， テスト：127件

結果

比較として、SBTシーケンス変換前と比較

- ソースコードのトークン数を300までに限定したものと、トークン数を無制限にしたもので実験
 - Transformerモデルの複雑さが入力・出力のシーケンス長に依存するため^[5]

	意味的類似度（平均）
SBT Seq（制限なし）	0.2389
Code Snippet（制限なし）	0.2290
SBT Seq（300）	0.1948
Code Snippet（300）	0.1991

タイトルを生成した結果

- SBTシーケンスに変換しても、意味的類似度に変化はなかった
 - どちらも類似度の値は低くなっている
 - 訓練データが少ない？

[5]A. Vaswani et al., “Attention is all you need,” NeurIPS,vol. 30, 2017.

結果の分析

例 1) 類似度 : 0.903

- 生成されたテキスト
 - Remove deprecated `GAUGE\_API\_PORT` variable
→ 非推奨の `GAUGE\_API\_PORT` 変数を削除する。
- 実際のIssue
 - Remove the use of GAUGE_API_PORT
→ GAUGE_API_PORTの使用を削除する。

例 2) 類似度 : 0.359

- 生成されたテキスト
 - Remove deprecated unread parameter from OPDSNavigationFeed
→ OPDSNavigationFeedから非推奨のunreadパラメータを削除しました。
- 実際のIssue
 - Add a services layer to the OPDS module.
→ OPDSモジュールにサービスレイヤーを追加する。

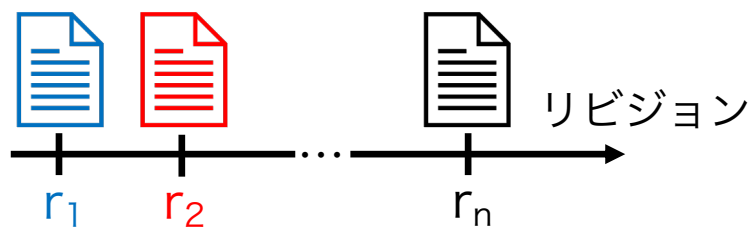
現在やっていること

TD Issueの本文には技術的負債について記述される

- TDが混入する前後での振る舞いの違いをIssueの本文として記述するのでは？

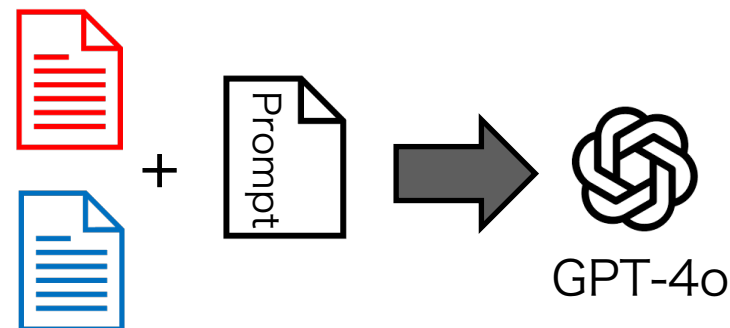
→ChatGPTにTD混入前後の振る舞いから本文を生成させる

1. TD混入前後のファイルを特定



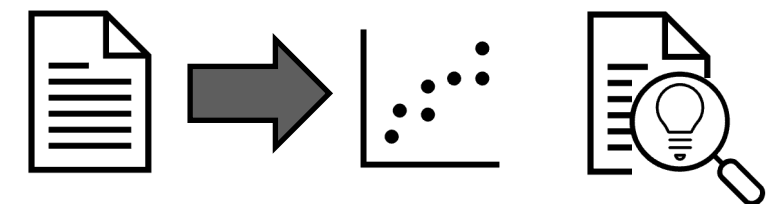
TDコード片が初めて出現した
リビジョンからTD混入コミット
を特定し、前後のファイルを取得

2. ChatGPTにプロンプトを入力



二つのコードをプロンプトに
組み込み本文を生成

3. 生成された本文の評価・分析



- 意味的類似度で評価
- 結果を目視し、
5段階で本文を評価する

実際に生成

- 生成された本文

- Potential Performance and Dependency Overhead Introduced by VideoMetadataReader

VideoMetadataReaderがもたらすパフォーマンスと依存性のオーバーヘッドの可能性

- Issueの本文

- A video metadata cache would be helpful so that the app doesn't have to compute duration/size/thumbnails every launch. Would need some way to handle external changes to files.

動画メタデータのキャッシュがあれば、アプリが起動のたびに再生時間／サイズ／サムネイルを計算する必要がなくなる。ファイルに対する外部からの変更を処理する何らかの方法が必要である。

- 類似度：0.474

補足：プロンプト

You are an AI assistant analyzing the differences in the behavior of two versions of a code.

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

1. the original version (before including technical debt)
2. the modified version (including technical debt)

Based on the differences in these behaviors, prepare a brief Issue body of no more than 65 tokens that explains what the change causes or implies. Focus on the functional, logic, and performance aspects affected by the change.

Format

Issue Body

Original code

Changed code (including technical debt)