

技術的負債に関連する 課題票分類手法の構築

オープンソースソフトウェア工学研究室

学籍番号：60219001

氏名：木村 祐太

技術的負債：Technical Debt^[1]

時間的制約によって導入される低品質な設計や実装

- 例) コードスメル：深刻な問題を抱えているコードの特徴

保守コスト増加を引き起こす^{[3][4][5][6]}

- 開発の約23～35%が余分な作業
- 既存のTDによる新たなTDの導入（TDの蓄積）

開発資源（人的，金銭的，時間的）は有限
TDの蓄積量を管理する必要がある

[1] Ward Cunningham. The wycash portfolio management system. SIGPLAN OOPS Mess., 4(2):29–30, 1992.

[3] Carolyn Seaman, Yuepu Guo, Nico Zazworka, Forrest Shull, Clemente Izurieta, Yuanfang Cai, and Antonio Vetr`o. Using technical debt data in decision making: Potential decision approaches. In Proceedings of the 3rd International Workshop on Managing Technical Debt, MTD, pages 45–48, 2012.

[4] Edith Tom, Ayb uKe Aurum, and Richard Vidgen. An exploration of technical debt. 86(6):1498–1516, 2013.

[5] Terese Besker, Antonio Martini, and Jan Bosch. The pricey bill of technical debt: When and by whom will it be paid? In Proceedings of the 33rd International Conference on Software Maintenance and Evolution, ICSME, pages 13–23, 2017.

[6] Terese Besker, Antonio Martini, and Jan Bosch. Technical debt cripples software developer productivity: A longitudinal study on developers 'daily software development work. In Proceedings of the 1st International Conference on Technical Debt, TechDebt, pages 105–114, 2018.

Self-Admitted Technical Debt

開発者によってTDが明示的に管理される事象

- SATD-Issue ^[7]
 - 課題票 (Issue) にTDラベルを割り当てて明示されるSATD
- ※ Issue : 不具合, 機能改善, 機能要求など様々な課題が管理される

Cannot use IDialogService in some services due to custom dialog service dependencies
#109056

commented on 21 Oct 2020

Trying to use `IDialogService` in `ExtensionService` or in its dependencies like `ExtensionManagementService` is causing workbench to explode because dialog service is depending on `Layout` that depends on `ExtensionService`.

Due to this dependency, `DialogService` cannot be used in many services. We should allow dialog service to be used by all workbench services and not introduce such a restriction.

Added `important` `debt` labels on 21 Oct 2020

Assigned `bpasero` and `sbatten` on 21 Oct 2020

commented on 21 Oct 2020

This came in via `1d4c7fa` and is probably a consequence of having the `window.dialogStyle: custom` setting even in native.

It does not seem to be the `ILayoutService` dependency btw, it seems to be these:

- `IThemeService`

Assignees

`sbatten`

Labels

`debt` `dialogs` `important` `insiders-released`

Projects

None yet

Milestone

November 2020

Linked pull requests

Successfully merging a pull request may close this issue.

`Refactor DialogService to use model and c...`



“debt”というTDラベルが
割り当てられている

引用 : <https://github.com/microsoft/vscode/issues/109056>

[7] Laerte Xavier, Fabio Ferreira, Rodrigo Brito, and Marco Tulio Valente. Beyond the code: Mining self-admitted technical debt in issue tracker systems. In Proceedings of the 17th International Conference on Mining Software Repositories, MSR, pages 137–146, 2020.

SATD-Issueの問題点と解決案

問題点

- SATD-Issueのデータ数が少ない → 研究を困難している
 - SATD-Issueを扱っているOSSプロジェクトが少ない



解決案

- 公開されているIssueのなかから、明示されていないが技術的負債と関連するIssueを抽出
 - SATD-Issueの分類に関する研究は存在しない

目的とアプローチ

目的

- SATD-Issue自動分類モデルの構築

アプローチ

1. SATD-Issueの特徴分析

- テキスト以外にモデル構築に利用できる特徴を明らかにする
- 仮説検定など定量的分析を利用

2. 分類モデルの評価実験

- 1. の分析で得られた特徴を用いて構築したモデルの評価
- 追加した特徴の分類への寄与の調査

データセット

リポジトリ	ラベル	SATD-Issue [件]	通常Issue [件]
microsoft/vscode	debt	1,612	102,292
influxdata/influxdb	kind/tech det	111	9,903
saleor/saleor	technical debt	106	2,623
nextcloud/server	technical debt	73	9,273

4つのOSSプロジェクトのデータを利用

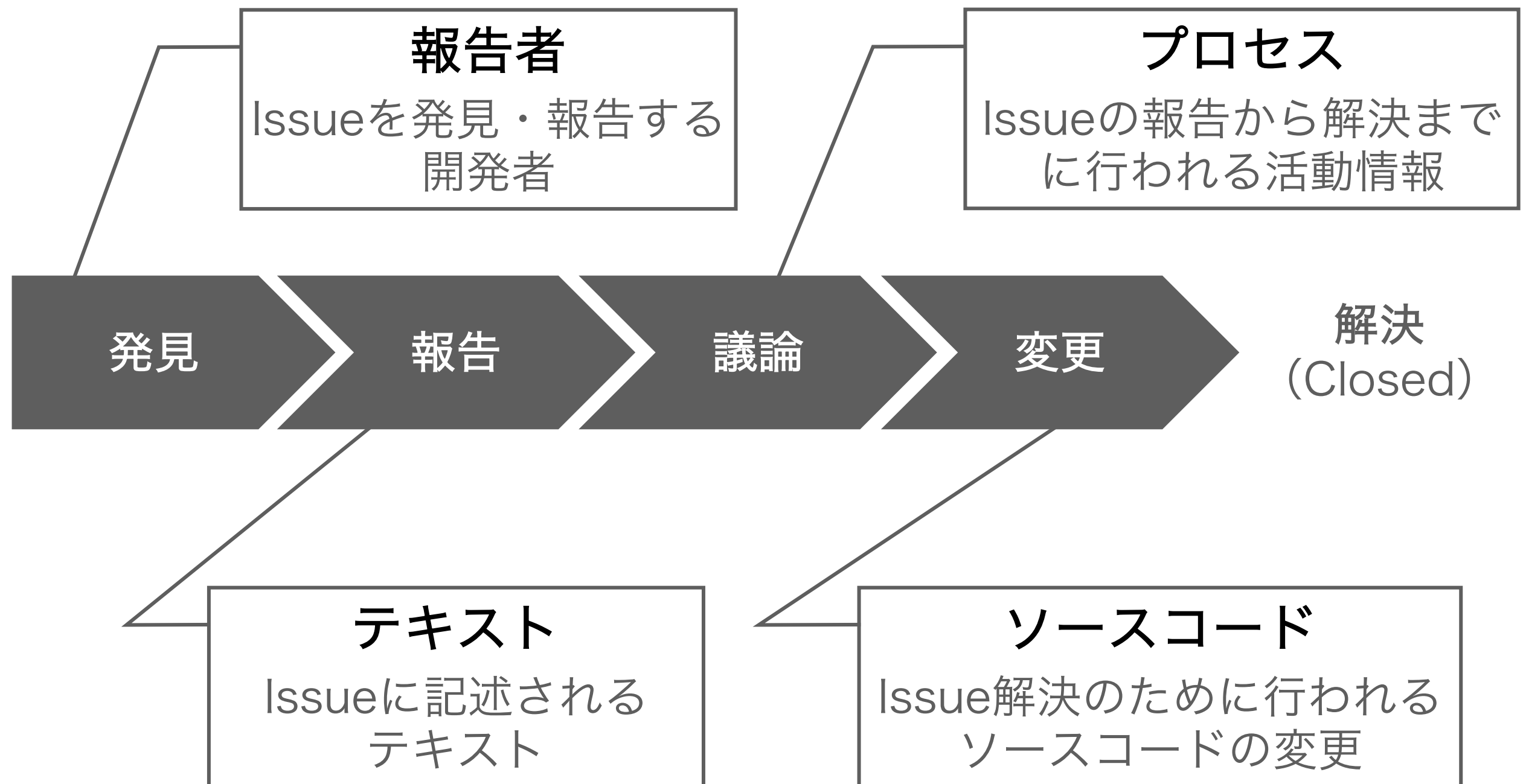
- GitHub（開発プラットフォーム）にホスティングされている
- SATD-Issueを慣習的に扱っている

Closed Issue（解決済みIssue）のみを対象

- 報告されてからしばらく時間が経過して、Issue内容を表すラベルが割り当てられることがある

SATD-Issueの特徴分析：検討

Issue発見から解決までの各工程に着目



SATD-Issueの特徴分析：方法

報告者，プロセス，ソースコードに関連する特徴を対象に以下の方法を実施

- 仮説検定と効果量測定

- 量的データ

- 仮説検定：マンホイットニーのU検定，効果量：Cliff's d

- 質的データ

- 仮説検定：カイ二乗検定，効果量：Cohen's w

- VIF統計量を用いた特徴削減

- 線形モデルを歪める多重共線性の高い特徴を使用しない

有意差 ($p < 0.01$) があり効果量が小程度以上かつ
VIF統計量が3未満の特徴を選択

評価実験：データセット作成

各特徴ごとに外れ値を含むデータを除外

- 各特徴が正規分布を仮定できないため、箱ひげ図を用いて外れ値を検出

$$\text{外れ値} < Q1 - 1.5 \times IQR, \quad Q3 + 1.5 \times IQR < \text{外れ値}$$

$Q1$: 第1四分位数, $Q3$: 第3四分位数, IQR : 四分位範囲

- 外れ値を1つ以上含むデータを除外

	特徴A	特徴B	特徴C
Issue1	外れ値	正常値	正常値
Issue2	正常値	外れ値	外れ値
Issue3	正常値	正常値	正常値



データを除外

評価実験：テキストの前処理

モデル学習のためにテキストのベクトル化が必要

1. 特定単語の置換

- URL, コード片, ファイル名, など

2. 単語の分割 (1-gram)

3. ストップワードとMarkdownタグの除去

- 分類のノイズとなるテキストの除去

4. 固有名詞とOut-of-Vocabularyの置換

- プロジェクト固有の表現として変換

5. 単語の正規化

- 小文字・数字統一, レマタイジング

6. Bag-of-WordsやTF-IDFを用いたベクトル化

評価実験：実験設定

データセット（外れ値除去後）

リポジトリ	SATD-Issue [件]	通常Issue [件]
microsoft/vscode	958	50,625
influxdata/influxdb	71	6,169
saleor/saleor	86	2,019
nextcloud/server	55	4,819

機械学習アルゴリズム

- ロジスティック回帰, ランダムフォレスト, SVM

評価方法

- 層化5分割交差検証を20回試行（計 100回）
- 平均Precision, Recall, F1値を評価指標として利用

評価実験：結果

		ロジスティック回帰			ランダムフォレスト			SVM		
		Pre.	Rec.	F1	Pre.	Rec.	F1	Pre.	Rec.	F1
vscode	BoW	0.983	0.837	0.904	1.000	0.862	0.925	0.982	0.895	0.936
	TF-IDF	0.988	0.831	0.903	1.000	0.889	0.941	0.995	0.903	0.947
influxdb	BoW	0.795	0.092	0.162	0.820	0.085	0.151	0.762	0.238	0.352
	TF-IDF	0.160	0.011	0.021	0.800	0.082	0.146	0.820	0.148	0.242
saleor	BoW	0.589	0.060	0.107	0.360	0.022	0.041	0.475	0.213	0.285
	TF-IDF	0.020	0.001	0.002	0.190	0.011	0.021	0.674	0.103	0.174
server	BoW	1.000	0.398	0.553	0.880	0.176	0.284	0.962	0.521	0.667
	TF-IDF	1.000	0.415	0.575	0.930	0.199	0.318	0.965	0.525	0.671

- F1値に注目すると、全てのプロジェクトにおいてSVMを用いた分類モデルが最も良い

評価実験：分類に寄与している特徴

アブレーションスタディの結果：microsoft/vscode

Model	SVM					
	Pre.		Rec.		F1	
T+R+P+C	*0.995	(+14.3%)	*0.903	(+44.1%)	*0.947	(+34.9%)
T+R+P	*0.996	(+14.4%)	*0.903	(+44.1%)	*0.947	(+34.9%)
T+R+C	*0.995	(+14.2%)	*0.795	(+33.4%)	*0.884	(+28.5%)
T+P+C	*0.958	(+10.6%)	*0.800	(+33.8%)	*0.872	(+27.3%)
T+R	*0.994	(+14.2%)	*0.801	(+34.0%)	*0.887	(+28.9%)
T+P	*0.957	(+10.5%)	*0.799	(+34.0%)	*0.871	(+27.2%)
T+C	0.850	(-0.2%)	0.458	(-0.4%)	0.594	(-0.4%)
T	0.852		0.462		0.598	

T：テキスト，R：報告者，P：プロセス，C：ソースコード

*： $p < 0.01$ （マンホイットニーのU検定）

- 報告者とプロセスに関する特徴が分類に寄与している
- ソースコードに関する特徴が分類に寄与していない

議論：テキストでの分類

テキストのみの分類モデル (vscode)

- 分類性能
 - Precision : 0.852, Recall : 0.462, F1 : 0.598
- テキストのみでもSATD-Issueの分類は可能

テキストの特徴表現や分類モデルのアーキテクチャの変更で分類性能が向上する可能性がある

テキストのみの分類モデルがより高い分類性能で分類可能になると
Issueが報告された時点でTDと関連するか判別できる



早期にTDの返済に取り掛かることが可能に

議論：報告者に関連する特徴

テキスト＋報告者の分類モデル (vscode)

- 分類性能
 - Precision : 0.994 (+14.2%) , Recall : 0.801 (+34.0%) , F1 : 0.887
- これまでのIssue報告数など, 報告者の開発経験を踏まえると分類性能が向上する
- ➡ 開発経験が豊富/乏しいに関係なく報告される内容が存在し, 開発経験の違いによってTDラベルの有無が生まれる

- 開発経験が豊富な開発者によって報告されるTD
- 開発経験が乏しい開発者によって報告されるTD

それぞれのTDの返済情報を調査することで
TDの優先順位付けに役立てることが可能に

まとめ

目的

- SATD-Issue自動分類モデルの構築

アプローチ

- SATD-Issueの特徴分析
- 分類モデルの評価実験

結果

- 最も分類性能が高いモデル
 - Precision : 0.995, Recall : 0.903, F1値 : 0.947
- 特徴ごとの分類への寄与
 - 報告者 ○, プロセス ○, ソースコード ×

博士後期課程での計画

研究背景：技術的負債（TD）について

TDの社会的問題

- 保守コストの増加 → 余分な工数の追加
 - ソフトウェア開発は、ユーザのニーズもしくはプロジェクトが理想とする機能を実装することが目的
 - 保守コストの増加は新規機能や機能アップデートの妨げとなる

TDの蓄積量を管理し開発資源を有効利用する必要がある

TDに関する研究の問題

- 研究者が問題と捉えるTDと開発者が問題と捉えるTDの隔たりが存在
 - 実際の開発では、既存のツールはあまり使われていない [8][9]

[8] Emerson Murphy-Hill, Chris Parnin, and Andrew P. Black. How We Refactor, and How We Know It. IEEE Transactions on Software Engineering, 38(1):5–18, 2012.

[9] Yaroslav Golubev, Zarina Kurbatova, Eman Abdullah AlOmar, Timofey Bryksin, and Mohamed Wiem Mkaouer. One Thousand and One Stories: A Large-Scale Survey of Software Refactoring. In Proceedings of the 29th European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE, pages 1303–1313, 2021.

研究背景：SATDへの着目

開発者によるTDの明示的管理

- Self-Admitted Technical Debt (SATD)
 - コメントやIssueを用いてTDを明示的に管理する事象

開発者が問題して管理するTDの理解が可能になる

SATD-Issue

- IssueにTDラベルを割り当てて明示するSATD
- さまざまな情報が取得が可能
 - TD混入の原因
 - TDの影響範囲（変更対象ファイル数など）
 - TDの返済方法（コードの修正パターンなど）

博士後期課程での研究計画（1）

TDの効率的な管理を支援するための研究

1年次・2年次

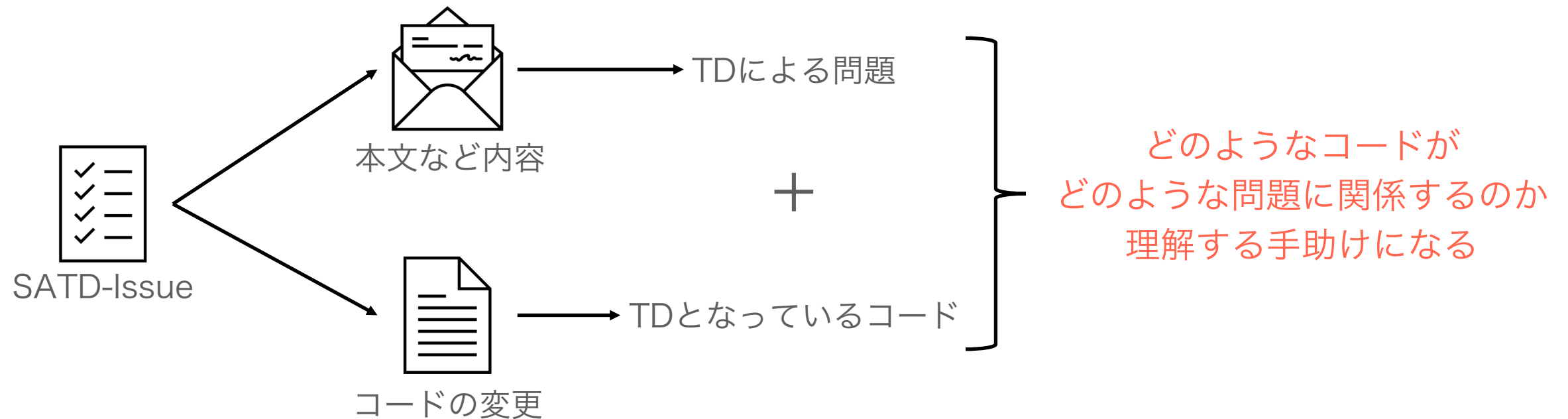
- TDの問題と対応するコードの関係分析
 - SATD-Issueの内容（本文など）に着目した調査
 - 分類モデルの構築
 - Issueで扱われている内容の分析
 - SATD-Issueの変更（コード）に着目した調査
 - TDとなっているコードのパターン
 - TDとなっているコードの変更パターン

3年次

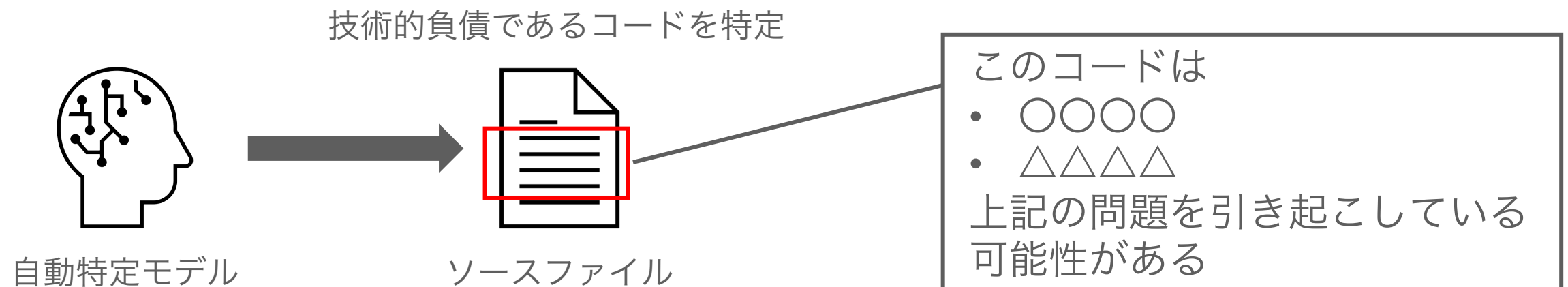
- TDとなるコードの自動特定に関する研究

博士後期課程での研究計画 (2)

TDとなるコードの自動特定に向けて



- 技術的負債となるコードを自動特定したとき, コードによって引き起こされている問題の理解を支援できる



SATD-Issueの内容に着目した研究

目的

- 開発者が問題として報告する技術的負債の内容を明らかにする

アプローチ

1. テキストベースの分類モデルの評価実験

- さまざまなテキストの特徴量表現を用いてモデルを構築
報告されたIssueのなかから

2. 分類結果に基づいたSATD-Issueの内容分析

- モデルによって分類できるSATD-Issueはプロジェクトに関係なく、一般的に技術的負債として扱われている内容と想定
- 分類できなかったSATD-Issueはプロジェクト固有の内容と想定

評価実験：データセット

報告者の開発経験に基づくデータフィルタリング

開発経験 (Issue, Pull Request, Commit) がある報告者のIssue数

リポジトリ	SATD-Issue		その他	
vscode	1,554 / 1,612	(96.4%)	57,309 / 102,292	(56.0%)
influxdb	111 / 111	(100.0%)	7,078 / 9,903	(71.5%)
saleor	105 / 106	(99.1%)	1,938 / 2,623	(73.9%)
server	63 / 73	(86.3%)	4,856 / 9,273	(52.4%)
合計	1,833 / 1,902		71,181 / 124,091	

- 開発経験がない開発者を除外する
 - Issue報告, Pull Request投稿, Commitの経験がない開発者を除外
 - SATD-Issueは開発経験が豊富な開発者によって報告される傾向にある

Category	Feature		p-value & Effect Size			
			vscode	influxdb	saleor	server
報告者 (7)	Experience		*M+	*S+	*S+	*M+
	OpenIssueNum		*L+	*M+	*L+	*L+
	OpenPullRequestNum		*L+	*L+	*L+	*L+
	rCommitNum		*L+	*S+	*S+	*L+
	Type	Member	*S+	N	*S+	*S+
		Contributor	*N	*N	N	N
		Collaborator	N	N	N	—



SATD-Issueの報告者はその他の報告者に比べて開発経験が有意に多い (有意差, 正の効果量)
→ 技術貢献が開発者は技術的負債の存在を知らない

評価実験：実験設計

テキストの特徴量表現

- Bag-of-Words
- TF-IDF

学習アルゴリズム

- ランダムフォレスト
- ロジスティック回帰
- サポートベクターマシン (SVM)

比較のベース
追加する特徴量表現やアルゴリズムは検討中

評価方法

- Leave-One-Out交差検証（修論からの変更）
 - 分類できた/できなかったIssueの分析を目的
- Precision, Recall, F1値

今後の予定：検討事項

ほかのテキストの特徴量表現の利用

- Word2Vec
 - テキストコーパスを学習したニューラルネットワークの中間層の重みを用いる単語埋込み技法
- ELMo
 - テキストコーパスを学習したBi-LSTMの中間層の重みを用いる単語埋込み技法

深層学習アルゴリズムの利用

- 自然言語処理では、単語の情報だけでなく文脈を情報として取り入れる目的でCNNやRNNが利用されることが多い

まとめ

目的

- 開発者が問題として報告する技術的負債の内容を明らかにする

アプローチ

1. テキストベースの分類モデルの評価実験
2. 分類結果に基づいたSATD-Issueの内容分析

今後の予定

- 2022年5月 SES 論文投稿予定

研究業績

| 2022年3月 SIGSE 研究発表

| 2022年3月 IPSJ論文誌 論文投稿（査読中）

補足資料

特徴分析の結果一覧

Category	Feature		p-value & Effect Size			
			vscode	influxdb	saleor	server
報告者 (7)	Experience		*M+	*S+	*S+	*M+
	OpenIssueNum		*L+	*M+	*L+	*L+
	OpenPullRequestNum		*L+	*L+	*L+	*L+
	rCommitNum		*L+	*S+	*S+	*L+
	Type	Member	*S+	N	*S+	*S+
		Contributor	*N	*N	N	N
		Collaborator	N	N	N	—
テキスト (*)	Title		—	—	—	—
	Description		—	—	—	—
プロセス (9)	TitleLen		*N	N	*N	*S-
	DescriptionLen		*M-	*M-	*N	*S-
	AssigneeNum		N	*S+	*S+	*S+
	SelfAssign		*S+	N	N	*N
	ParticipantNum		*M-	*L-	*S-	N
	CommentNum		*S-	*S-	*S-	*S+
	pCommitNum		*L+	N	N	N
	ChangeFileNum		*L+	N	N	N
	ResolutionTime		*M+	*M+	*S+	*L+
ソースコード (14)	AddedClassNum		*S+	—	—	*S+
	DeletedClassNum		*S+	—	—	N
	AddedFunctionNum		*S+	—	—	N
	DeletedFunctionNum		*S+	—	—	N
	AddedTotalLine		*M+	—	—	*M+
	DeletedTotalLine		*L+	—	—	*S+
	AddedLineOfCode		*S+	—	—	S
	DeletedLineOfCode		*M+	—	—	N
	AddedLineOfComment		*S+	—	—	N
	DeletedLineOfComment		*M+	—	—	N
	AddedEssential		*S+	—	—	N
	DeletedEssential		*S+	—	—	*S+
	AddedMaxNesting		*S+	—	—	S+
	DeletedMaxNesting		*S+	—	—	N

*：有意差あり，\：検定不可，分析対象外

N：効果量なし，S：小程度の効果量

M：中程度の効果量，L：大程度の効果量

+：正の効果量，-：負の効果量