

技術的負債返済による 保守コスト削減効果モデル構築 に向けた実証的調査

オープンソースソフトウェア工学研究室
M1 60210032 木村 祐太

技術的負債 (Technical Debt)

- Cunningham がリファクタリングの重要性を説くために導入したメタファ^[1]
- 短期的には利益が得られるが、将来的な保守で高いコストを支払わなければならない設計や実装
 - 開発者の能力, リリース, 市場競争などによって作為的,あるいは, 無作為的に引き起こされる
- 技術的負債は大規模システムの長期運用における保守運用の世界的な課題

[1] Cunningham, W., 1992. The wycash portfolio management system. Proc. Obj. Oriented Programm. Syst. Lang. Appl. 4 (2), 29–30.

技術的負債のイメージ

- 技術的負債の混入（貸入）と返済を図で説明

余力があれば

技術的負債の影響調査

- 従来研究をまとめる

Table 9

TD measurement approaches and related studies.

Category	Description	Studies
Calculation model	Calculate TD through mathematical formulas or models	S11, S14, S16, S17, S20, S22, S37, S39, S50, S51, S55, S58, S61, S62, S70, S72, S83
Code metrics	Calculate TD using metrics of source code	S3, S7, S25, S39, S83
Human estimation	Estimate TD according to experience and expertise	S34, S72
Cost categorization	Estimate various types of the cost of handling the incurred TD [S80]	S80
Operational metrics	Indicate TD using quality metrics of product operation [S74]	S74
Solution comparison	Calculate the distance between the actual solution and the optimal solution [S23]	S23

Table 12

TD monitoring approaches and related studies.

Category	Description	Study
Threshold-based approach	Define thresholds for TD related quality metrics, and issue warnings if the thresholds are not met [S22]	S22
TD propagation tracking	Track the influences of TD through dependencies between other parts of a system and the parts of the system that contains TD [S39]	S39
Planned check	Regularly measure identified TD and track the change of the TD [S7]	S7
TD monitoring with quality attribute focus	Monitor the change of quality attributes that detrimental to TD, such as stability [S5]	S5
TD plot	Plot various aggregated measures of TD over time and look at the shape of the curve to observe the trends [S72]	S72

従来研究の課題

- 技術的負債を貸入しているときの影響調査
- コードの技術的負債を対象
- 技術的負債を返済することで得られる効果がわからない

技術的負債の種類をまとめた表

本研究の目的

- 技術的負債の返済による保守コスト削減効果を統計的に分析調査しモデル化を図ること
- 今回の研究スコープ
 - 技術的負債の返済前後の保守コスト比較分析
 - 対象とする技術的負債
 - Self-Admitted Technical Debt Issue (SATD-I) ^[?]

[?] Laerte Xavier, Fabio Ferreira, Rodrigo Brito, and Marco Tulio Valente. 2020. Beyond the Code: Mining Self-Admitted Technical Debt in Issue Tracker Systems. In Proceedings of the 17th International Conference on Mining Software Repositories (MSR '20). Association for Computing Machinery, New York, NY, USA, 137–146.

Issue（課題管理票）

- GitHub^[?]（共同開発環境）に実装されている機能
- 開発している製品の不具合や機能改善要求など様々な内容を管理

ステータス

Open：未解決
Closed：解決済

Single text edits never go with a group set to the UndoRedoService #107677

New issue

 Closed

jrieken opened this issue on 29 Sep · 3 comments

報告内容



jrieken commented on 29 Sep

Member ...

@alexdimas thanks, `DEBUG` helps me figure out why these edits are in two different groups. The bulk edit ([#101789 \(comment\)](#)) includes a text edit and a notebook edit, and we pass in the same `undoRedo` group. However when applying the text edit, since `tasks.length === 1`, it does not use the `undoRedo` group passed in (if there are more than one text edit, they then are put in the same group).

[vscode/src/vs/workbench/contrib/bulkEdit/browser/bulkTextEdits.ts](#)
Lines 215 to 239 in 4aa873b

```
215     const validation = this._validateTasks(tasks);
216     if (!validation.canApply) {
217         throw new Error(`${validation.reason.toString()} has changed in the meantime`);
218     }
219     if (tasks.length === 1) {
220         // This edit touches a single model => keep things simple
221         for (const task of tasks) {
222             task.model.pushStackElement();
223             task.apply();
224             task.model.pushStackElement();
225             this._progress.report(undefined);
226         }
227     }
```

I'm wondering if we should allow pass in `undoRedoGroup` in `task.model.pushStackElement()` (and all its inner `pushStackElement` call)

Originally posted by @rebornix in [#101789 \(comment\)](#)

Assignees

 jrieken
 alexdimas

Labels

 debt

Projects

None yet

Milestone

September 2020

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

2 participants



内容に関連するラベル

[?] <https://github.com>

SATD-I^[?]

Self-Admitted Technical Debt Issue

- 開発者が技術的負債に関連するラベルを付与し報告する Issue のこと

Single text edits never go with a group set to the UndoRedoService #107677

New issue

技術的負債を表すラベル

jrieken commented on 29 Sep

@alexdimas thanks, `DEBUG` helps me figure out why these edits are in two different groups. The bulk edit (#101789 (comment)) includes a text edit and a notebook edit, and we pass in the same undoRedo group. However when applying the text edit, since `tasks.length === 1`, it does not use the undoRedo group passed in (if there are more than one text edit, they then are put in the same group).

```
vscode/src/vs/workbench/contrib/bulkEdit/browser/bulkTextEdits.ts
Lines 215 to 239 in 4aa873b

215     const validation = this._validateTasks(tasks);
216     if (!validation.canApply) {
217         throw new Error(`${validation.reason.toString()} has changed in the meantime`);
218     }
219     if (tasks.length === 1) {
220         // This edit touches a single model => keep things simple
221         for (const task of tasks) {
222             task.model.pushStackElement();
223             task.apply();
224             task.model.pushStackElement();
225             this._progress.report(undefined);
226         }

```

I'm wondering if we should allow pass in undoRedoGroup in `task.model.pushStackElement()` (and all its inner `pushStackElement` call)

Originally posted by @rebornix in #101789 (comment)

Assignees

- jrieken
- alexdimas

Labels

- debt

Projects

None yet

Milestone

September 2020

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

2 participants

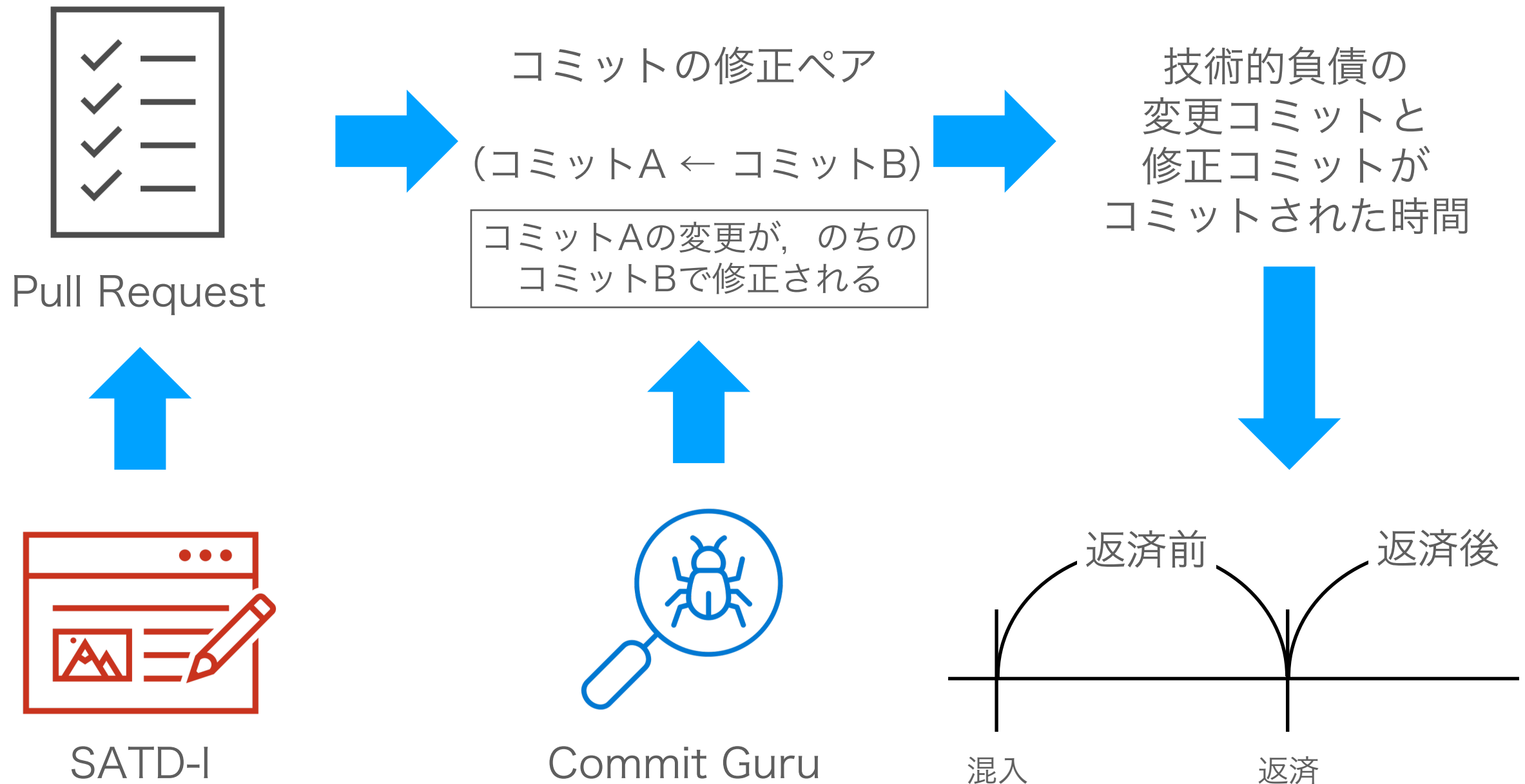
[?] Laerte Xavier, Fabio Ferreira, Rodrigo Brito, and Marco Tulio Valente. 2020. Beyond the Code: Mining Self-Admitted Technical Debt in Issue Tracker Systems. In Proceedings of the 17th International Conference on Mining Software Repositories (MSR '20). Association for Computing Machinery, New York, NY, USA, 137–146.

期間の設定方法

1.SATD-Iに関連する
修正コミットの紐付け

2.技術的負債に関連する
コミットの修正ペアの特定

3.技術的負債の
返済前後期間の設定



Pull Request (変更提案)

- Issue 同様, GitHub に実装されている機能
- 新たな機能案やバグの修正案などの投稿を管理

ステータス

Open : 未解決
Merge : 変更済

Remove terminal renderers #78529

Merged

Tyriar merged 11 commits into master from tyriar/69865_remove_renderers on 7 Aug 2019

提案に関する議論

Tyriar commented on 6 Aug 2019

Fixes #69865

Remove terminal renderers

Tyriar added this to the August 2019 milestone on 6 Aug 2019

Tyriar self-assigned this on 6 Aug 2019

Some clean up

Tyriar assigned alexr00 on 6 Aug 2019

Tyriar commented on 6 Aug 2019

@alexr00 I disabled CustomExecution by throwing exceptions and changing types to any, can you remove the rest in this branch?

@GabeDeBacker are we blocked on remove terminal renderers or are we good to merge this when ready?

New issue

+121 -853

Reviewers

alexr00

Assignees

Tyriar

alexr00

Labels

None yet

Projects

None yet

Milestone

August 2019

Linked issues

Successfully merging this pull request may close these issues.

Remove deprecated terminal renderer APIs

2 participants

関連する Issue のリンク

SATD-I と修正コミットの紐付け

- SATD-I が紐づいている Pull Request を抽出
- それらの Pull Request に紐づいているコミットを技術的負債に関連する修正コミットとして扱う

Remove terminal renderers #78529

New issue

Merged Tyriar merged 11 commits into master from tyriar/69865_remove_renderers on 7 Aug 2019

Conversation 6 Commits 11 Checks 11 Files changed 16

Pull Requestに
紐づいているコミット

Commits on Aug 5, 2019

Remove terminal renderers

Tyriar committed on 6 Aug 2019

Some clean up

Tyriar committed on 6 Aug 2019

Merge remote-tracking branch 'origin/master' into tyriar/69865_remove...

Tyriar committed on 6 Aug 2019

Remove checks for process manager existence in terminal instance

Tyriar committed on 6 Aug 2019

Commits on Aug 6, 2019

Remove CustomExecution

alexr00 committed on 6 Aug 2019

Merge remote-tracking branch 'origin/master' into tyriar/69865_remove...

Tyriar committed on 6 Aug 2019

Merge branch 'master' into tyriar/69865_remove_renderers

Tyriar committed on 7 Aug 2019

Merge remote-tracking branch 'origin/master' into tyriar/69865_remove...

Tyriar committed on 7 Aug 2019

Merge remote-tracking branch 'origin/master' into tyriar/69865_remove...

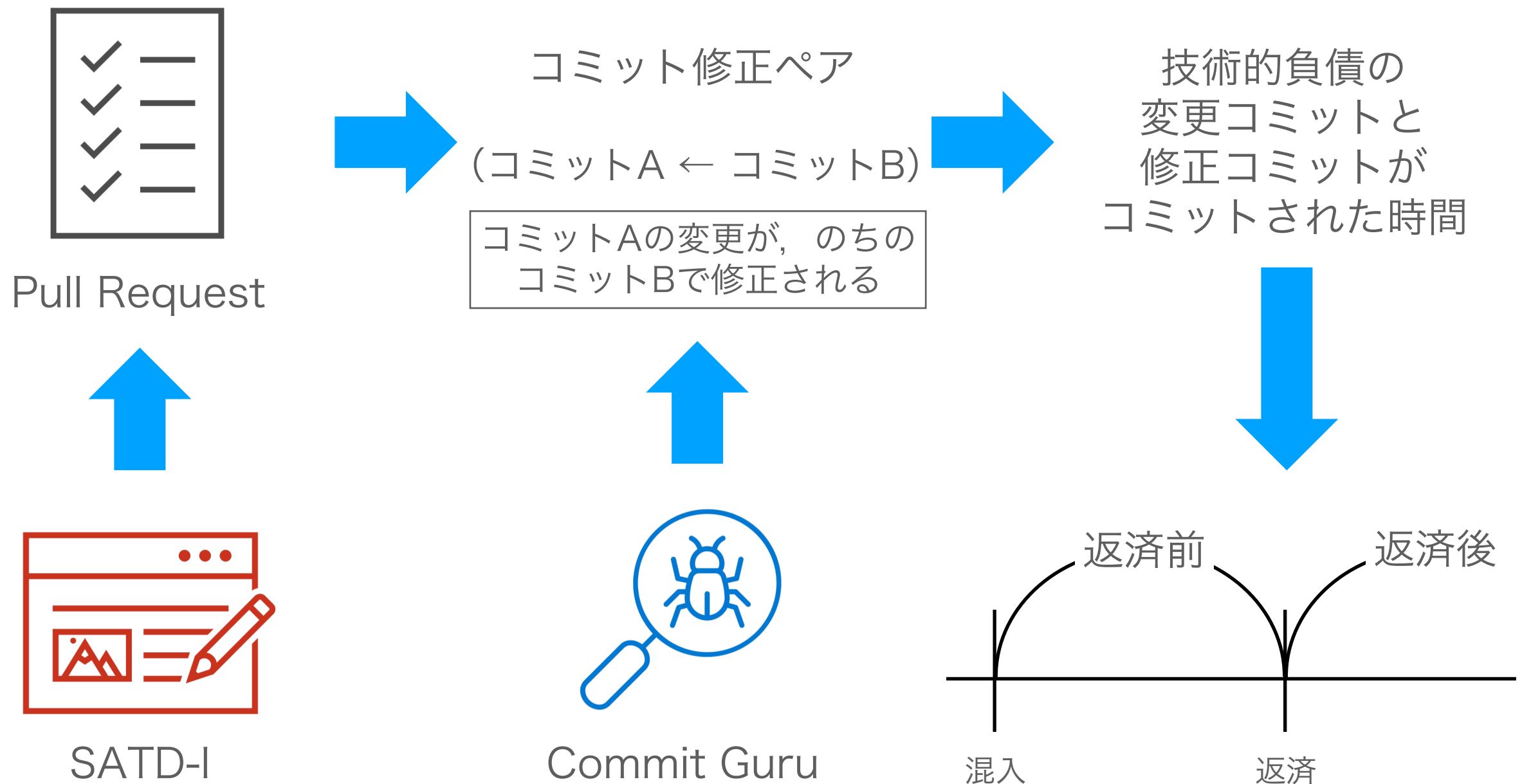
Tyriar committed on 7 Aug 2019

期間の設定方法

1.SATD-Iに関連する
修正コミットの紐付け

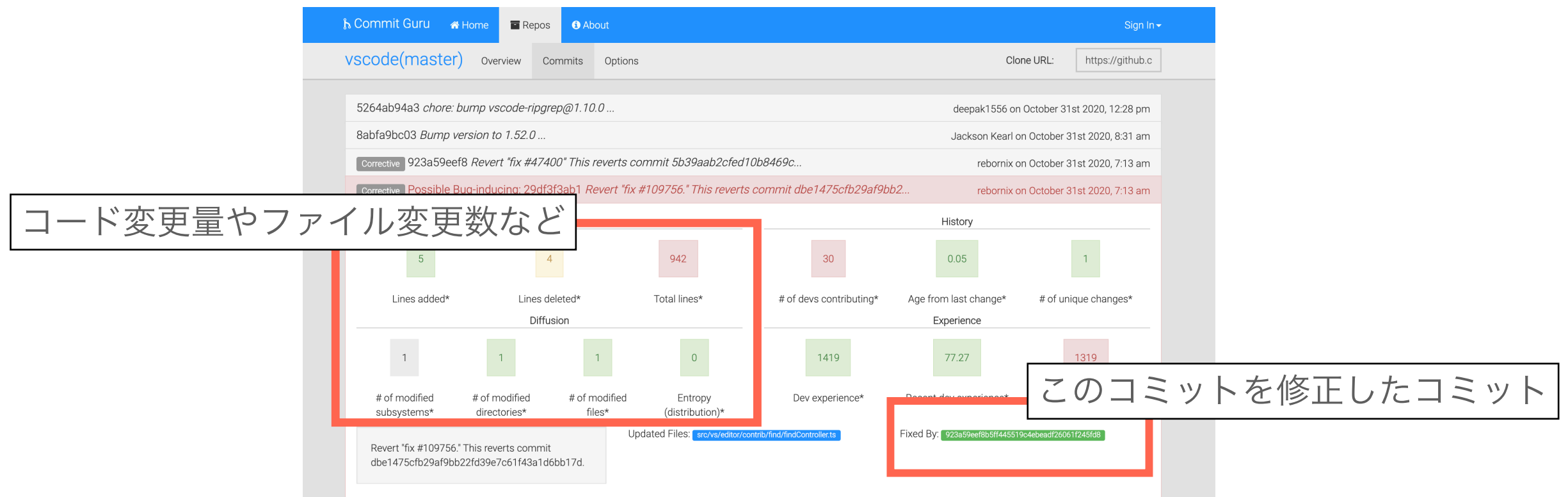
2.技術的負債に関連する
コミットの修正ペアの特定

3.技術的負債の
返済前後期間の設定



Commit Guru^[?]

- ソフトウェアリポジトリ分析ツール
 - コミット修正ペアの抽出
 - コミットのコード変更量, ファイル変更数 など
 - バグを含んでいるコミットの特定

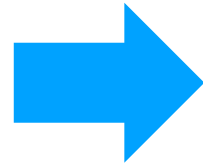


[?] Christoffer Rosen, Ben Grawi, and Emad Shihab. 2015. Commit guru: analytics and risk prediction of software commits. In Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015). Association for Computing Machinery, New York, NY, USA, 966–969.

技術的負債の修正ペアの特定



Commit Guru



(コミットA, コミットB)
修正ペア

1.で特定した技術的負債の修正コミット
を含んだ修正ペアを抽出



(混入日時, 返済日時)

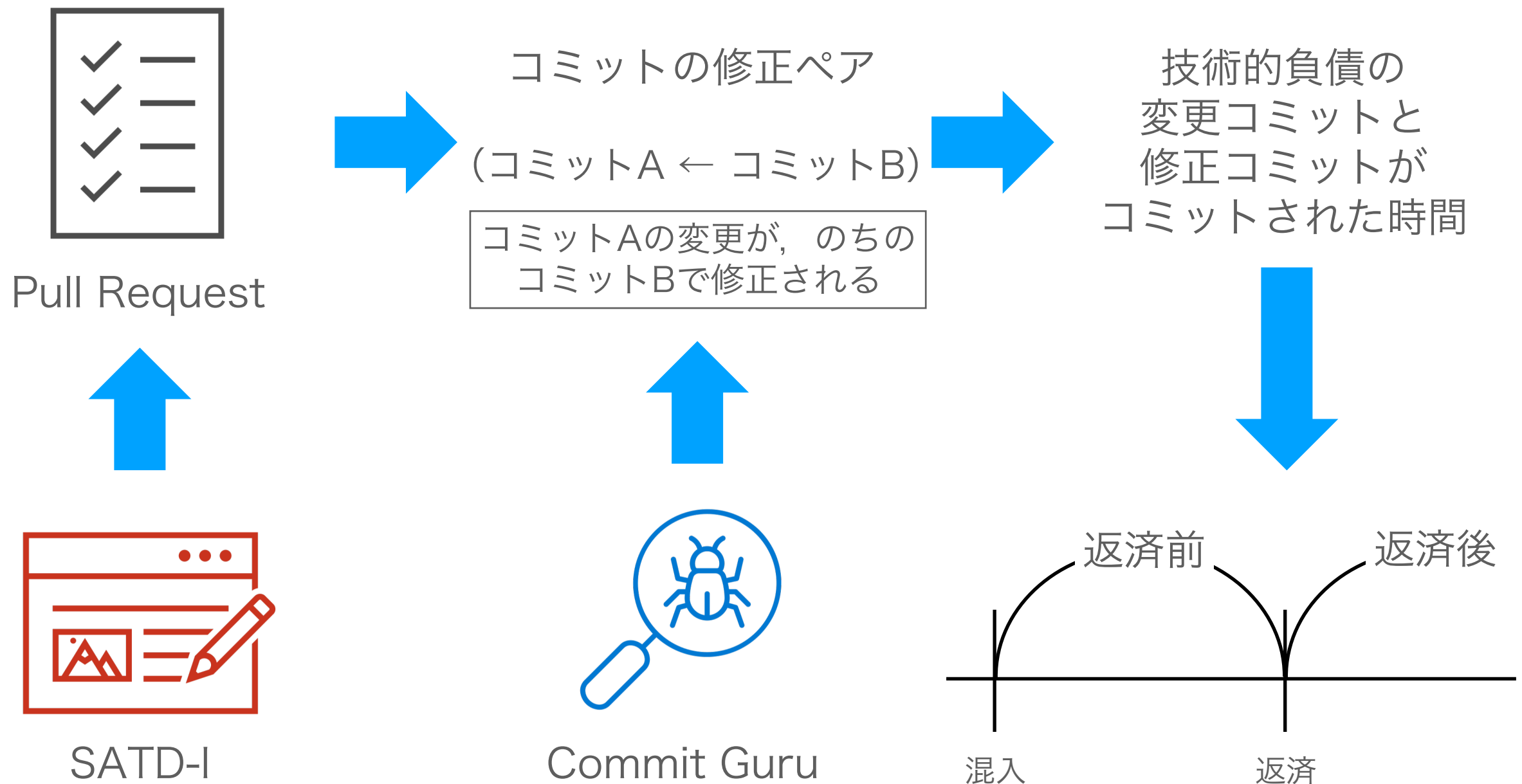
抽出された修正ペアの
コミットAが投稿された時間を混入日時
コミットBが投稿された時間を返済日時

期間の設定方法

1.SATD-Iに関連する
修正コミットの紐付け

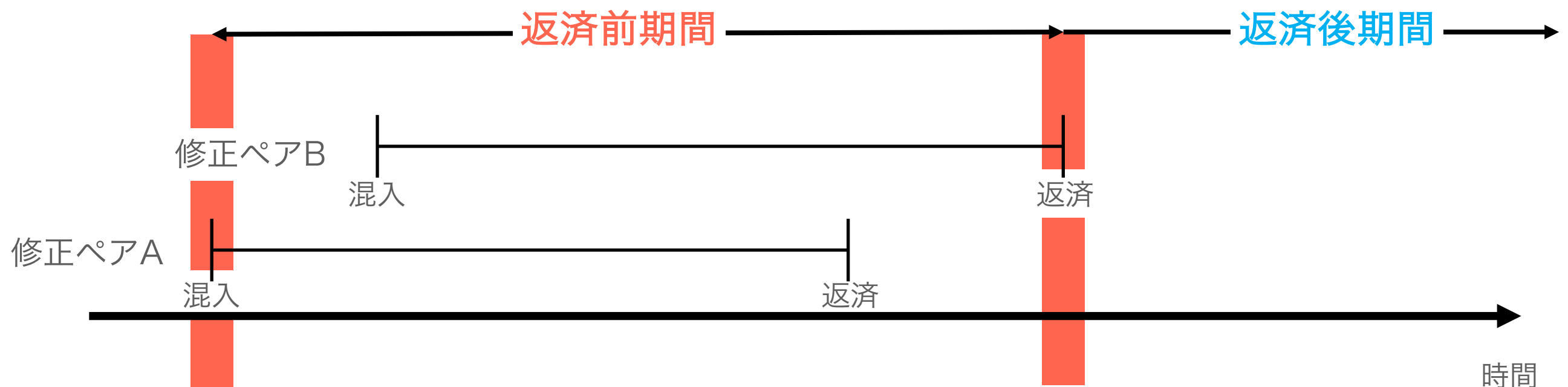
2.技術的負債に関連する
コミットの修正ペアの特定

3.技術的負債の
返済前後期間の設定



技術的負債の返済前後期間の設定

- 1つの SATD-I に対して複数の返済（修正）が行われる場合がある
- 返済前期間は，関連する修正ペアの中で最も早い混入日時を開始とし最も遅い返済日時を終了とする
- 返済前期間の終了以降を返済後期間とする



分析

- 各 SATD-I における返済前後の保守コストを比較
- SATD-I に関連するファイルに変更を加えたコミットのみを抽出し保守コストを算出
- 算出した各期間の保守コストの倍率で評価

$$\text{倍率} = M_b / M_a$$

M_b : 返済前期間の保守コスト

M_a : 返済後期間の保守コスト

比較する保守コスト項目

保守コスト項目	説明
バグ発生率 (バグ数/コミット数)	各期間中のコミットにバグが含まれていた割合
コミット頻度 (コミット数/日)	各期間中の1日あたりの平均コミット回数
オーナーシップ (コミット数/開発者)	各期間中に開発者1人あたりの平均コミット数
同時ファイル変更数	各期間中の1コミットあたりの平均同時ファイル変更数
コード変更量	各期間中の1コミットあたりの平均コード変更数
バグ修正時間	各期間中に混入したバグが修正されるまでにかかった平均時間

対象データセット^[?]

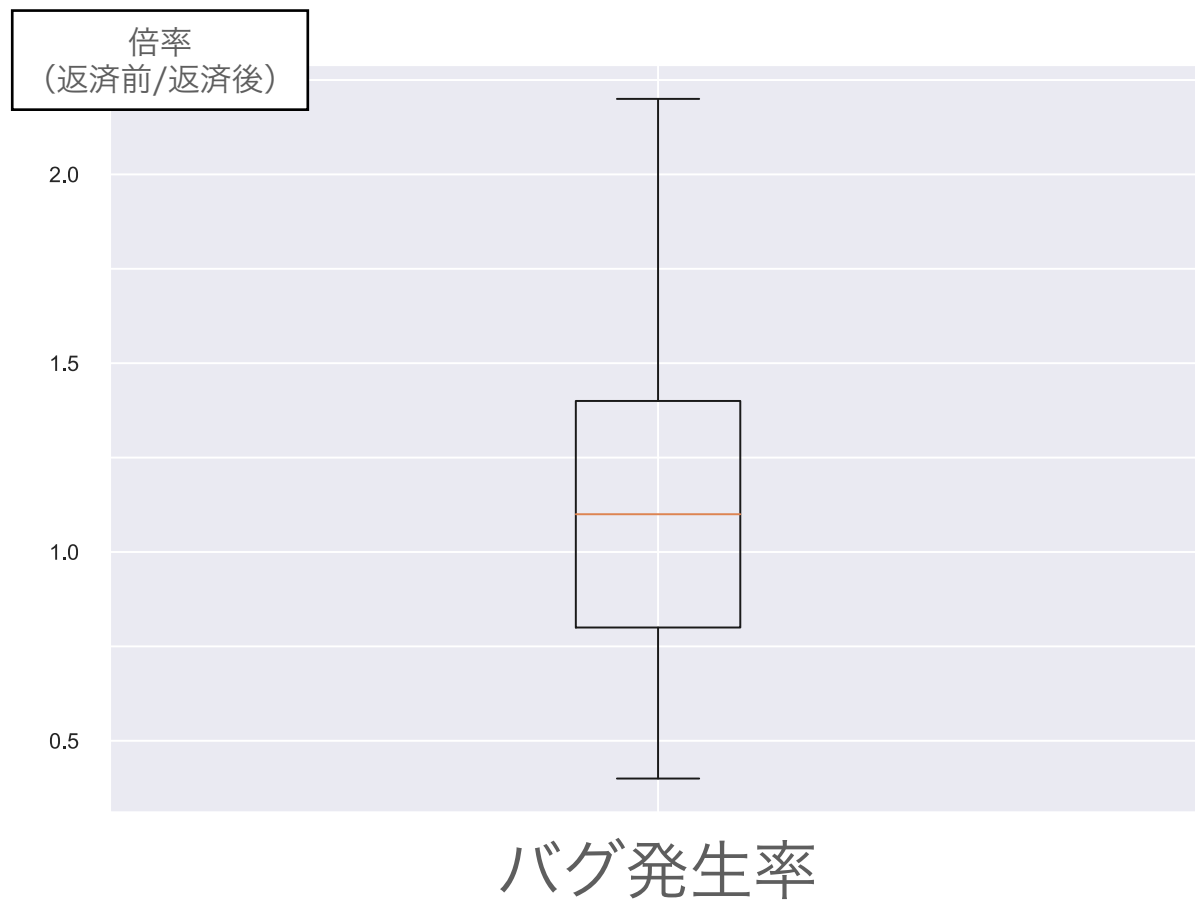
- 解決済（Closed）された SATD-I を対象

プロジェクト名	期間	コミット	Closed SATD-I
InfluxDB	2013年4月11日 ～ 2020年8月15日	33,906	99
Saleor	2013年2月12日 ～ 2020年8月18日	17,013	164
VScode	2015年11月13日 ～ 2020年8月18日	69,557	1,418

[?] Laerte Xavier, Fabio Ferreira, Rodrigo Brito, and Marco Tulio Valente. 2020. Beyond the Code: Mining Self-Admitted Technical Debt in Issue Tracker Systems. In Proceedings of the 17th International Conference on Mining Software Repositories (MSR '20). Association for Computing Machinery, New York, NY, USA, 137–146.

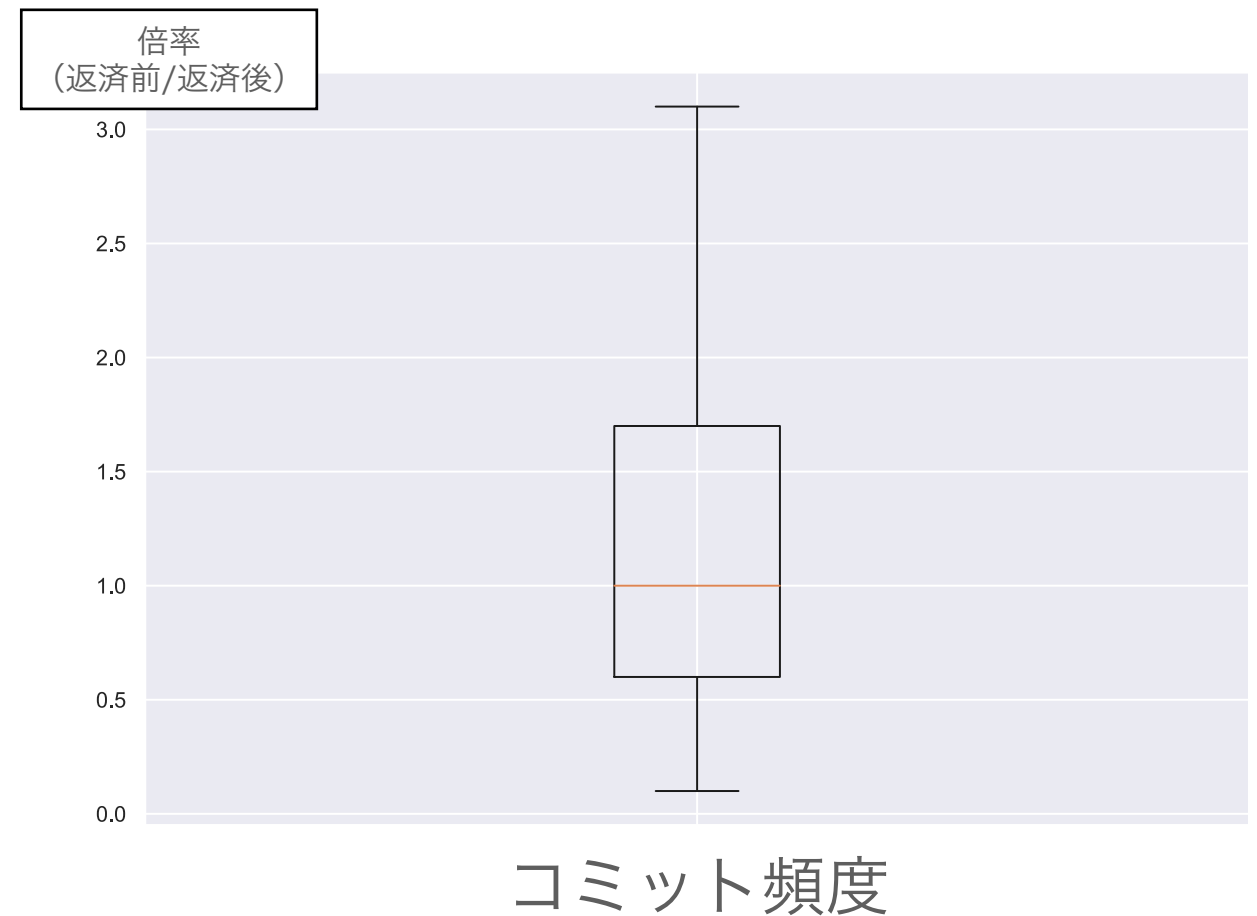
比較結果（１）

バグ発生率とコミット頻度



中央値 > 1.0

返済前の方が
バグが発生しやすい

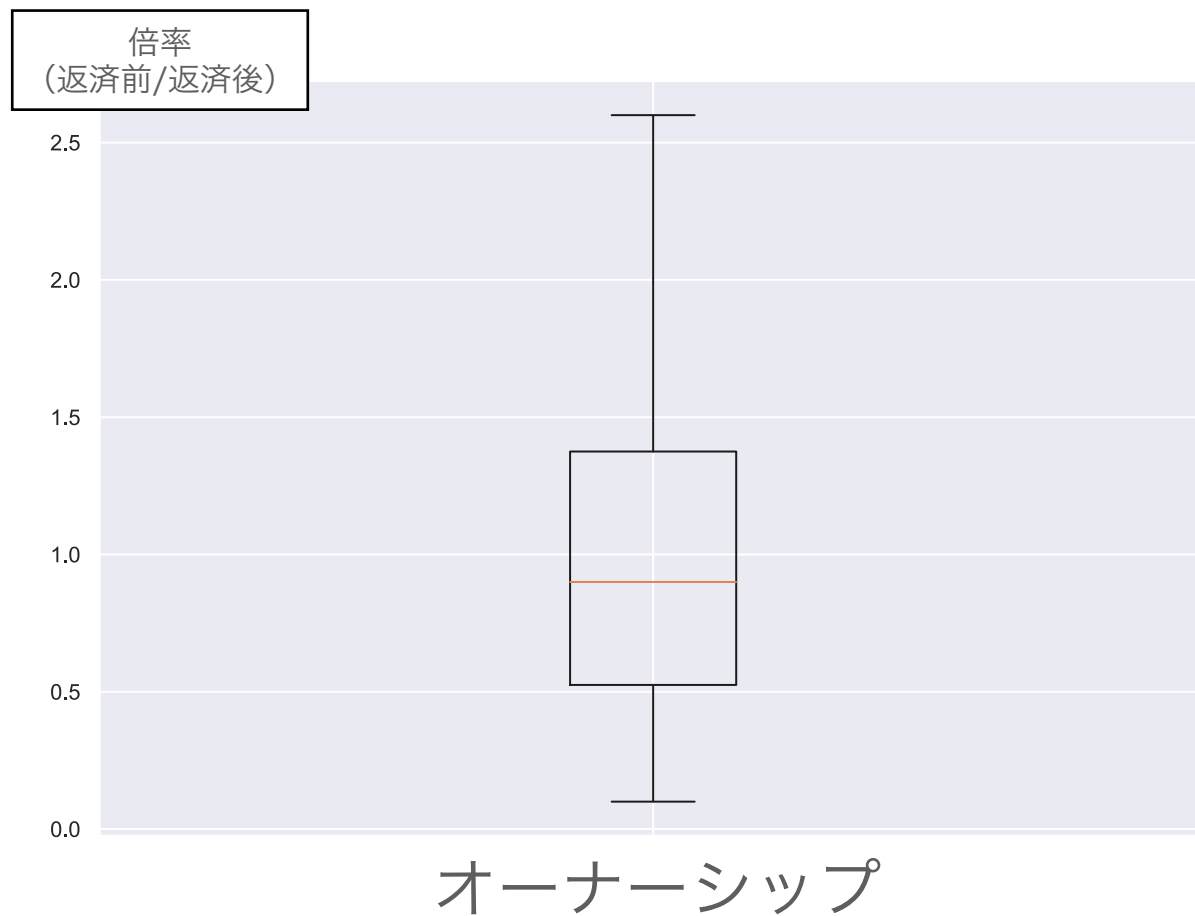


中央値 = 1.0

返済前後で
コミット頻度は変わらない

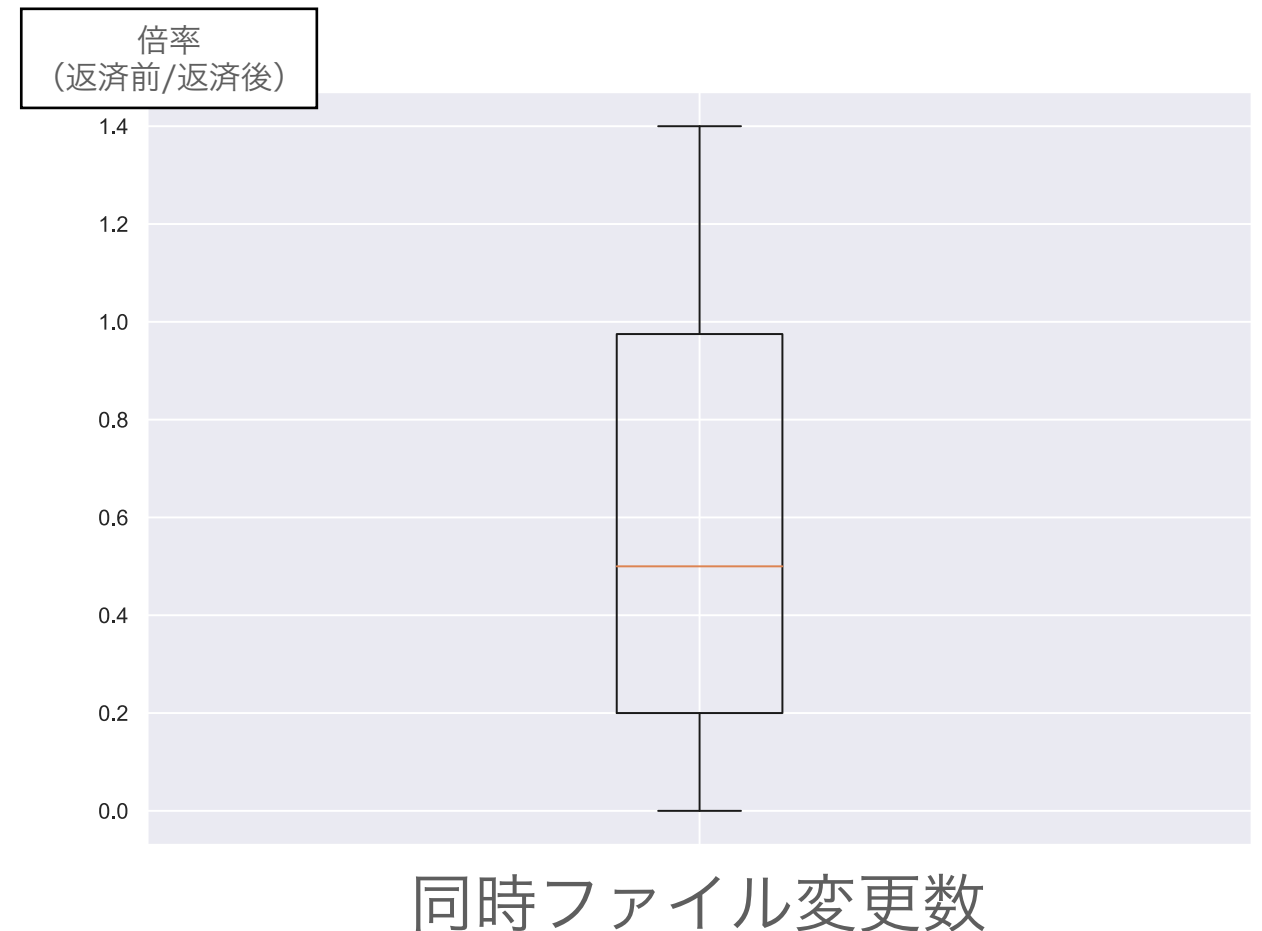
比較結果 (2)

おーなシップと同時ファイル変更数



中央値 < 1.0

返済後の方が
開発者1人あたりのコミット数が多い

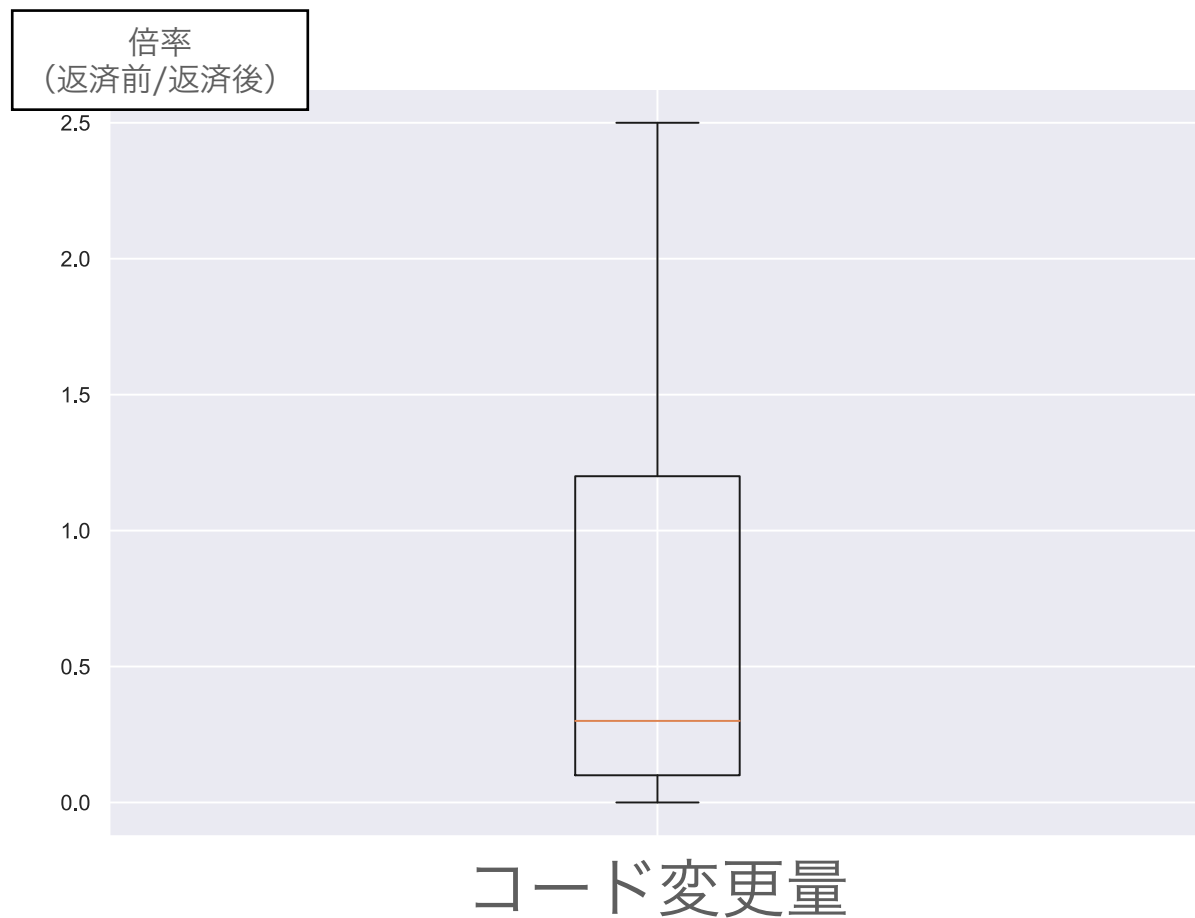


中央値 < 1.0

返済後の方が
同時に変更するファイル数が多い

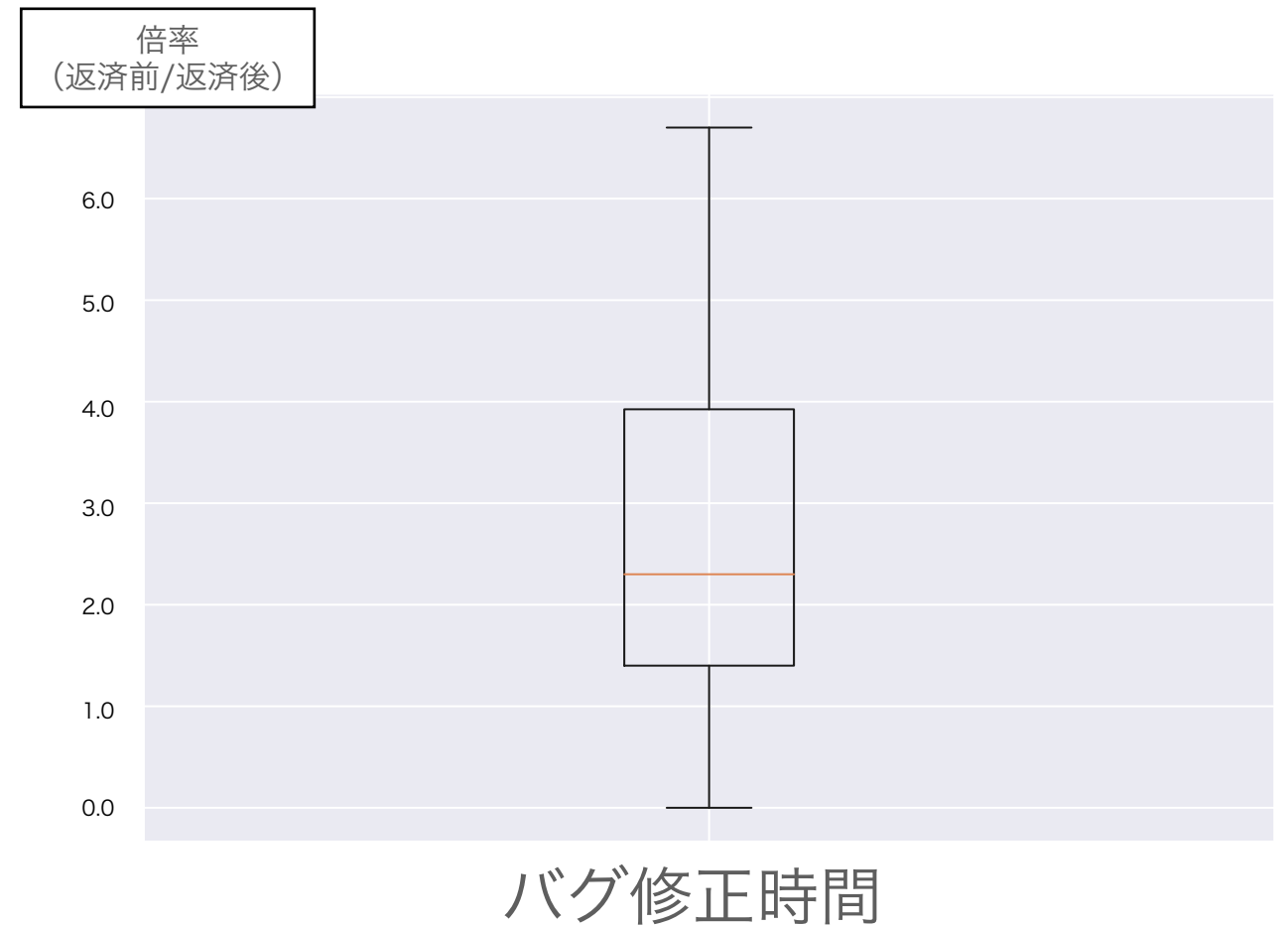
比較結果（3）

コード変更量とバグ修正時間



中央値 < 1.0

返済後の方が
コード変更量が多い



中央値 > 1.0

返済前の方が
バグ修正に時間がかかる

まとめ

- 返済前の方が高い

- バグ発生率, バグ修正時間

技術的負債 (SATD-I) は
バグを引き起こしやすく修正に時間がかかる

- 返済後の方が高い

- オーナーシップ, 同時ファイル変更数, コード変更量

オーナーシップ, 同時ファイル変更数,
コード変更量などの規模が増加していることから,
技術的負債 (SATD-I) の返済後は変更が容易になる

今後の計画

- SATD-I 以外の技術的負債についての分析
 - 従来研究ではツールによって技術的負債を検出
- SATD-I を扱っているプロジェクトを調査し、分析の規模を広げる