

静的コード解析ツールが検出するTDとSATD-issueの調査

2021/11/17

発表者: 松尾 和輝

Technical Debt

- Technical Debt(TD)

- 時間がかかるが良い実装を行わず短期的で雑な実装を行なったために修正が必要なコード

- Self-Admitted TD(SATD)

- 開発者自身がソースコードコメントや課題票を用いてTDであることを明示して管理しているケース

- SATD-Code

- SATDのうち、ソースコードコメントを用いて管理しているケース

- SATD-Issue

- SATDのうち、課題票を用いて管理しているケース

背景

- TDは、締め切りや開発コストの削減等で混入し、プログラムの品質の低下や保守コストの増加をもたらすため、早期に発見し取り除く必要がある
- TDの発見には、一般的に静的コード解析ツールが用いられており、自動で効率的に行えるため発見の手間を減らすことができる
- 近年では、SATD-CodeやSATD-Issueのような開発者がTDであることを明示し管理しているケースが出てきている

問題

- ツールで検出できるTDの他, SATD-CodeやSATD-IssueのようなTDが出てきた現在, どのTDから修正を行えば良いか分かっていない
 - すぐに修正した方が良いTDを見逃してしまったり, 修正が遅れたために保守コストの増加を引き起こす恐れがある

目的とアプローチ

- 目的

- 静的コード解析ツールが検出するTDとSATD-Issueの違いを明らかにする
 - 今回の研究ではSATD-Issueと静的コード解析ツールが検出するTDのみを対象とする

- アプローチ

- 解析対象はvscode
- SATD-issueに紐付くコミット(変更後のリビジョン)を静的コード解析ツール(SonarQube)を用いて解析
- 解析した結果を用いて分析

SonarQube

- コード内の「バグ」, 「脆弱性」, 「Code Smell」などを予め設定されたルールに基づいて検出することができる静的コード解析ツール
- 自分でルールを追加して解析を行うことができる
- Java, JavaScript, TypeScript, C, C++, C#, HTML, PHP, CSS等のファイルを解析することができる

データ収集

- SATD-Issueを慣習的に扱っているプロジェクトである“vscode”を解析対象とする
- vscode内の“debt”というタグの付いた**1612**件のIssueを収集し、それに紐づくリビジョンを特定する
- 特定したリビジョンは**2219**件

データ収集

- 変更前のリビジョンの特定
 - 変更後のリビジョンを”checkout”コマンドを用いることにより、一つ前のリビジョンに戻すことができ、変更前のリビジョンを特定することができる
- SonarQubeによる解析
 - 特定した変更前後のリビジョンをSonarQubeで解析
 - 変更前後それぞれで1398リビジョンが解析成功
- SATD-IssueはSonarQubeで検出できるようなTDを除去できているのかを調査

やったこと，わかったこと等

- 1612件のSATD-Issueに対して，SATD-Issueに紐づくリビジョンが一つ以上解析できた846件(52.5%)のSATD-Issueを分析対象とする
- 変更前後で一つ以上TDを除去できたSATD-Issueの数は77件(9.1%)だった

今後の予定

- 分析方法を考える
 - SATD-Issueとツールで検出できるTDの修正にかかる時間を比較する
 - 時間が短い方が開発者にとってより早く修正すべきTDである可能性