コードレビューにおけるソースコード修正理由の自動説明に向けて

上田 裕己[1,a)]　伊原 彰紀[2,b)]　石尾 隆[1,c)]　松本 健一[1,d)]

概要：オープンソースソフトウェアプロジェクトには多くのソースコード変更提案が不特定多数の開発者から投稿される．レビューアと呼ばれる複数の開発者は，投稿されたソースコードに含まれる問題を取り除くためにソースコードを検証し修正を投稿者に依頼するが，その作業には膨大なコストが必要となる．この問題に対してソースコード修正を自動提案する手法の研究が行われているが，自動提案した内容が有用であるか否かはレビューアが確認する必要があり，レビューアがソースコードと修正理由を理解するコストが増加することが考えられる．本研究では，この理解コストを軽減するために，ルールに従ったソースコード修正を行うべき理由の自動説明を行うシステムを提案する．

1. はじめに

ソーシャルコーディングプラットフォームである GitHub では，プルリクエストという機能を利用して多くのソースコード変更提案がプロジェクトに投稿されており，レビューアと呼ばれる開発者が変更を受け入れるべきか否かを検証し，問題がある場合は投稿者に修正を依頼する．すべての問題が取り除かれるまで，レビューアはソースコードを複数回検証を行う必要がある．レビューアのコスト削減を目的として，pylint [*1]や pmd [*2] といった自動ソースコード修正手法が多くのソフトウェア開発に導入されている [1]．一方，自動修正提案は言語仕様に基づくものであり，プロジェクトに適したルールが提案されるとは限らない．そのため，提案が有用であるか否かをレビューアが確認する必要があり，レビューアのソースコード理解コストが増加することが考えられる．

本研究で提案する自動修正提案システムは，ソースコードの修正提案だけでなく，修正を行う理由を説明することで，レビューアと投稿者のコード理解コストの削減を目指す．

[1] 奈良先端科学技術大学院大学
[2] 和歌山大学
[a)] ueda.yuki.un7@is.naist.jp
[b)] ihara@sys.wakayama-u.ac.jp
[c)] ishio@is.naist.jp
[d)] matumoto@is.naist.jp
[*1] pylint: `https://www.pylint.org/`
[*2] pmd: `https://pmd.github.io/`

2. ソースコード修正推薦の利用シナリオ

ソースコードの保守性を向上させるために，多くのプロジェクトがソースコードの修正方法を提案するツールをコードレビュー活動に採用している [2]．しかし，頻繁に行われたソースコードの修正パターンだとしても，プロジェクトや流行，重要度によっては提案された修正を適用する必要がない場合が考えられる．そこで，本研究で提案するソースコード自動修正提案システムには，修正例を提案する**修正機能**と修正例に修正理由説明コメントを追加する**修正説明機能**を実装する．これにより，投稿者が GitHub を通じて投稿したプルリクエストに対して，修正を提案すると同時に，その修正をレビューアが適用すべきか否かの判断を補助するコメントを提示する．

本研究で提案するシステムの処理の流れを**図 1** に示す．提案システムの修正機能と修正説明機能は以下の処理を行う．

(1.) 修正機能が投稿されたソースコードに対し，修正例を出力する．
(2.) 修正機能が修正説明機能に修正例をクエリとして渡す．
(3.) 修正説明機能がソースコード修正履歴から修正コードの差分と差分に付随したコメントのペアを収集する．
(4.) 修正説明機能によって修正機能で出力した修正例に合う説明を出力する．
(5.) 修正機能が修正への説明を修正説明機能から受け取る．
(6.) 修正機能が修正例とその説明を実装者に返す．

修正機能は，すでに多くの自動ソースコード修正手法が提

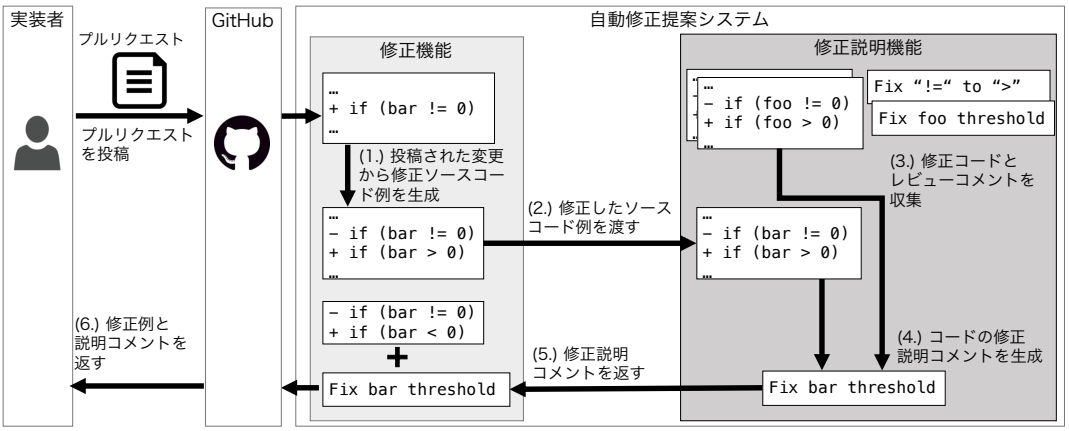

図 1 自動修正推薦の概要

Listing 1 レビューコメント "Why not simply return configValue?" への修正対応例

```
// 修正前のソースコード
public String toConfigValue() {
    return name().toLowerCase(Locale.ROOT).
        replace('_', '-');
}
// 修正後のソースコード
public String toConfigValue() {
    return configValue;
}
```

案されており，我々の研究グループではコードレビューにおける IF 文の自動修正提案を行うボットシステムを実装した [3]．本研究では修正説明機能の実装に着目する．

3. 修正説明機能の実装

利用するデータセット: コードレビューの一連の作業を管理するコードレビュー管理システムである Gerrit Code Review [*3]を対象にケーススタディを行う．先行研究では Gerrit API を利用することで Gerrit Code Review におけるソースコード変更前後の差分を示した `Diff` ファイルを取得した [3]．Paixao らは Gerrit Code Review で変更されたソースコードとそのレビューコメントを収集したソースコードを公開している [4]．JGit プロジェクトで実際に投稿されたレビューコメントの事例を Listings 1 に示す．

本研究では Paixao らが公開するレビューコメントに示されたファイル名，ファイル行数と先行研究で収集したソースコードを紐付けることで修正理由説明コメントを生成する．

生成アプローチ: ソースコード，またソースコードの修正を自然言語形式にまとめる手法は多くの研究が行われている．Jiang らは `Diff` ファイルからコミットメッセージ (e.g. Added X to readme, Remove redundant commands in travis config) を機械翻訳を通して生成する手法を提案した [5]．また，Moreno らは構文の変化からリリースノートを生成する手法を提案した．本研究ではレビューコメントに近い形式である，コミットメッセージ形式の出力を行う．本研究では，既存研究を利用して出力可能なパターンを抽出し，該当ソースコードの対して修正コメントを生成する．

[*3] Gerrit Code Review: `https://code.google.com/p/gerrit/`

4. おわりに

本論文では，レビューアによる説明コストの軽減を目的として，ソースコードの修正提案に修正理由の説明コメントを追加するアプローチを提案した．本研究を進めることで，低コストでコード理解の補助ができると考えている．

ワークショップでは提案した修正例に対して他にどのような出力が有用であるかを議論したい．

謝辞 本研究の一部は，JSPS 科研費 JP18H03221，JP17H00731，JP18KT0013 及びテレコム先端技術研究支援センター SCAT 研究の助成を受けたものです．

参考文献

[1] Panichella, S., Arnaoudova, V., Di Penta, M. and Antoniol, G.: Would static analysis tools help developers with code reviews?, *Proc. SANER*, pp. 161–170 (2015).

[2] Allamanis, M., Barr, E. T., Bird, C. and Sutton, C.: Learning Natural Coding Conventions, *Proc. FSE*, pp. 281–293 (2014).

[3] Ueda, Y., Ihara, A., Hirao, T., Ishio, T. and Matsumoto, K.: How is IF Statement Fixed Through Code Review? - A case study of Qt project -, *Proc. IWPD*, pp. 207–213 (2017).

[4] Paixao, M., Krinke, J., Han, D. and Harman, M.: CROP: Linking Code Reviews to Source Code Changes, *Proc. MSR*, pp. 46–49 (2018).

[5] Jiang, S., Armaly, A. and McMillan, C.: Automatically Generating Commit Messages from Diffs Using Neural Machine Translation, *Proc. ASE*, pp. 135–146 (2017).