7L-05

障害内容に応じたソースコード修正方法の分析

大森 楓己 † 伊原 彰紀 ‡ 松田 和輝 § 才木 一也 ¶
和歌山大学 † 和歌山大学 ‡ 和歌山大学 § 和歌山大学 ¶

1 はじめに

多くのソフトウェア開発では，脆弱性や実行速度の低下などのソフトウェアの障害解決に膨大な時間を投じている．開発者が障害の原因となる欠陥を修正したとしても環境に応じて再発することがあり，欠陥の修正は豊富な実装経験や障害解決の経験を有する開発者であっても困難な作業である．

従来研究では，ソースコードに含まれる欠陥を自動修正する手法に関する研究が多数行われている．既存の自動修正手法は多様な障害の修正履歴からパターンを抽出し，膨大な変更案を推薦している．多くの障害に共通する修正方法を明らかにし，Sobreira らは欠陥を含むソースコードの修正履歴に対して変更前後のソースコード（修正パッチ）から生成した抽象構文木に基づき欠陥修正パターンを抽出する手法を提案した [1]．膨大な修正案の中には適切な修正方法を含むことが多い一方，開発者が適切な修正案を選出するために時間的コストを要する．

本研究では，自動修正における修正案の絞り込みに向けて，障害内容に応じた修正パターンの違いを調査する．特に，ソフトウェア開発において膨大に報告されるソフトウェアの性能効率性に関する障害と障害全体における修正パターンの違いを調査する．

An analysis of source code modification according to the nature of the failure

† Fuki Omori, Wakayama University
‡ Akinori Ihara, Wakayama University
§ Kazuki Matsuda, Wakayama University
¶ Kazuya Saiki, Wakayama University

2 分析

本研究では，GitHub に公開されているソフトウェアのリポジトリの中で，スター数上位 500 件のリポジトリに投稿された変更履歴（プルリクエスト）を対象とする．調査する修正パッチは，Java 言語のソースコードファイルの修正を対象とする．その結果，分析対象プロジェクトに導入された 78,054 件のプルリクエストを収集した．

さらに，性能効率性に関する修正パッチは，従来研究 [2] と同様にプルリクエクストのタイトルまたはディスクリプション（説明文）に「performance」「slow」「hang」のいずれかを含む 520 件を対象とする．修正パターンの抽出には，Madeiral らが作成したパターン自動抽出ツール[*1]を用いる．性能効率性に関する障害と障害全体の修正パッチで，9 種類の修正パターンの抽出頻度を比較し，考察する．

3 分析結果

表 1 は，従来研究において頻出した修正パターンと，本研究の分析において計測した性能効率性に関する障害に対する修正パターンの頻度を示す．表は，性能効率性の修正パターンの頻度順に示す．従来研究では Cond. Block（条件処理を追加または削除）が最も多く抽出され，次いで Expr. Fix（式を変更），Wraps-with（条件処理内外へ処理を移動）が多く抽出された．一方，性能効率性に関する障害の修正では

[*1] https://github.com/lascam-UFU/automatic-diff-dissection

表 1: 性能効率性に関する障害の修正パターン

順位	修正内容	PR 数（割合 [%]）	従来研究の順位	従来研究の PR 割合 [%]
1	Copy/Paste	292(56.2)	7	12.2
2	Wrong Ref.	284(54.6)	5	17.8
3	Cond. Block	254(48.8)	1	42.8
4	Wraps-with	221(42.5)	3	27.3
5	Expr. Fix	186(35.8)	2	32.9
6	Const Chage	161(31.0)	8	4.8
7	CodeMoving	103(19.8)	9	1.8
8	Miss NullCheck	67(12.9)	6	12.7
9	SingleLine	32(6.2)	4	24.8

Copy/Paste（同一の変更を異なる箇所で反復）が最も多く抽出された修正パターンであり，次いで Wrong Ref.（変数またはメソッドの参照を変更），Cond. Block が多く抽出された．したがって，性能効率性に関する障害の修正パターンは，障害全体の修正パターンと異なる特徴があることが示唆される．

Copy/Paste が性能効率性に関する障害の修正パターンとして多数抽出された理由として，性能効率性に関する障害の修正は機能内容に関係しないため，複数箇所で同様の修正が行われることが多いと考える．また，性能効率性に関する障害の修正パターンとして多数抽出された Wrong Ref. は，大半がメソッド参照を変更する修正であった．ボトルネックになることが多い複数回使用されるメソッドの修正が多く，修正過程でメソッド呼出しの変更が多く発生したと考えられる．

本分析を通して，多様な障害から抽出した修正パターンと，性能効率性のみに対する修正パターンには違いがあることが示唆されるため，障害別の自動生成ツールの構築が有効となると期待する．

4 おわりに

本論文では，ソフトウェア開発において膨大に報告されるソフトウェアの性能効率性に関する障害に対して適用された修正パッチを対象として，障害別に修正パターンを調査した．分析の結果，性能効率性に関する障害では障害全体と適用される修正パッチの特徴が異なり，コピーペーストのように同一の変更を異なる箇所で繰り返す修正や，メソッド参照を変更する修正が多いことを確認した．今後は，性能効率性に関する障害以外の障害内容についても修正パターンの特徴を明らかにし，障害内容に応じたソースコード修正作業の効率化を目指す．

謝辞

本研究は JSPS 科研費 18H03221 の助成を受けたものです．

参考文献

[1] Sobreira, V., Durieux, T., Madeiral, F., Monperrus, M. and Maia, M. A.: Dissection of a Bug Dataset: Anatomy of 395 Patches from Defects4J, *Proceedings of the International Conference on Software Analysis, Evolution and Reengineering (SANER'18)*, pp. 130–140 (2018).

[2] Zaman, S., Adams, B. and Hassan, A. E.: Security versus Performance Bugs: A Case Study on Firefox, *Proceedings of the 8th Working Conference on Mining Software Repositories (MSR'11)*, pp. 93 – 102 (2011).