5K-03

欠陥混入コミット特定のための開発履歴へのファジング適用

小菅 遥生 †　　藤原賢二 ‡
東京都市大学 †　　東京都市大学 ‡

1　はじめに

ソフトウェアのデバッグを行う際，ソースコードや問題を引き起こす入力など様々な情報を使用する．特に，欠陥が混入されたコミットが分かる場合，修正すべきソースコードの範囲を狭めることができ，デバッグの効率化に繋がると考えられる．ソフトウェア工学の分野では欠陥混入を特定する手法として SZZ [1] が広く使われているが，この手法は欠陥が修正された後に，その欠陥が混入したコミットを特定するための仕組みであり，デバッグへの直接の活用は難しい．一方，ソフトウェアの不具合を自動的に検出する手法として近年ファジングが注目されている．ファジングとは，検査対象のソフトウェアに問題を引き起こしそうなデータを大量に入力し，その応答や挙動を監視することで不具合を検出する手法である．ファジングをソフトウェアに対して自動的に適用するためのファジングツールが実装されており，代表的なツールとして AFL（American Fuzzy Lop）[2] がある．

本研究では，AFL により検出されたエラーを引き起こす入力を，ソフトウェアの過去のコミットに入力することで，欠陥混入コミットを特定する手法を提案する．本稿では，提案手法の有効性を確認するために 2 つの OSS に対して提案手法を適用した．その結果，欠陥が修正されたコミットが判明していなくても，欠陥混入コミットを SZZ と同様に特定できることを確認した．

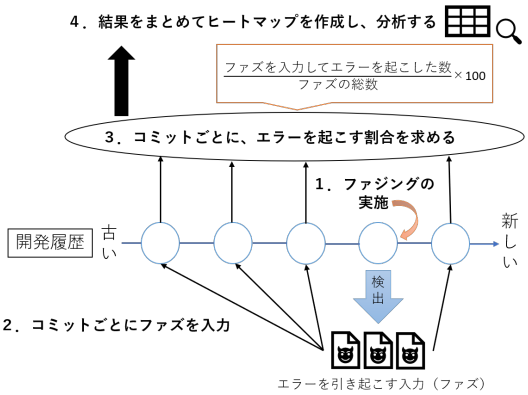

図 1　手法の概要図

2　提案手法

本研究では，ファジングによって検出したエラーを引き起こす入力を，同じ欠陥を含む過去のコミットに入力しても同様のエラーを起こすという考えに基づき，ファジングを用いた欠陥混入コミットの特定手法を提案する．なお，本研究でファジングを実施するツールとして AFL を採用する．手法の手順を図 1 に示す．提案手法ではまず，開発履歴から 1 つずつコミットを選択してファジングを実施し，コミットごとにファズと呼ばれるエラーを引き起こす入力を検出する．次にコミットごとに検出したファズを別のコミットに対して入力していく．そして，ファズを検出したコミットと，ファズを入力するコミットの組ごとに，ファズが実際にエラーとなる割合を求めていく．そして，それぞれ組ごとに求めた割合から，エラーとなる割合が高くなるほど濃い赤色になるヒートマップを作成する．最後にヒートマップを分析し，エラーを起こす割合が急増した時点のコミットを「不具合混入コミット」，急減した時点のコミットを「修正コミット」として特定する．

Fuzzing Software Development History for Detecting Fix Inducing Changes
† Kosuge Haruki, Tokyo City University
‡ Kenji Fujiwara, Tokyo City University

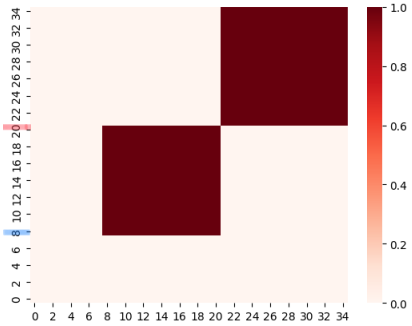

図 2　ctags のヒートマップ

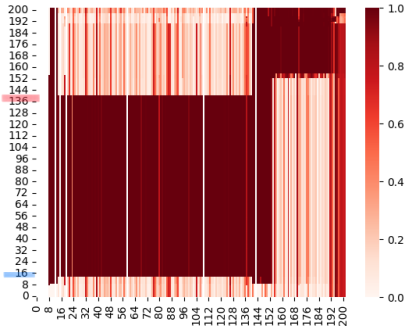

図 3　indent のヒートマップ

3　適用実験

提案手法の有効性を検証するために，AFL によって不具合が検出された実績がある OSS のうち ctags [*1] と indent [*2] に対して提案手法を適用した．適用したコミットはそれぞれ，ctasg は 36，indent は 202 コミットであった．

ctags のリポジトリは，OpenBSD の主要なコマンドを全て管理しているリポジトリであったため，適用対象を ctags に関係するファイルに変更があったコミットのみに限定した．これらのコミットに対してファジングをそれぞれ 10 分間適用し，ファズを得た．

indent への適用では，著者らの環境でビルドができなかった最も古い 4 コミットを適用対象から除外した．なお，ファジングはそれぞれ 6 時間適用しファズを得た．

3.1　ヒートマップを用いた欠陥混入の特定

提案手法を適用し，作成したヒートマップをそれぞれ，図 2（ctags），図 3（indent）に示す．提案手法において本質的ではないが，未来のコミットに対してもファズを入力した．縦軸，横軸の数値はそれぞれ，最新のコミットを 0 番目とし，以降古いコミットほど番号が大きくなる．この図では，横軸のコミットがファズを検出したコミットに対応し，縦軸がファズを入力したコミットに対応する．ヒートマップを分析し，どちらも縦軸の赤線で示すコミットでエラーを起こす割合が急増し，青線で割合が急減している．つまり，赤線で示すコミットで欠陥が混入し，青線で示すコミットで修正されたことがわかる．

3.2　SZZ による検証

前節で特定した欠陥混入コミットが正しいかどうかを確認するため，欠陥修正コミットに対して SZZ を適用し，SZZ の結果と一致するかどうかの確認を行った．その結果，ctags は，SZZ の結果と提案手法の結果が完全に一致した．indent は，SZZ を適用すると不具合混入コミットが 2 件特定されたが，その片方が提案手法の結果と一致した．

4　おわりに

本研究では，ファジングにより得られたエラーを引き起こす入力を，ソフトウェアの過去のコミット入力することで，欠陥混入コミットを特定する手法を提案した．そして 2 つの OSS に提案手法を適用し，欠陥混入コミットが特定できることを確認した．

謝辞　本研究の一部は JSPS 科研費 JP21H03416 の助成を受けた．

参考文献

[1] Śliwerski, J., Zimmermann, T. and Zeller, A.: When do changes induce fixes?, pp. 1–5 (2005).

[2] 情報処理推進機構：ファジング実践資料（AFL 編）(2020).

[*1] https://github.com/openbsd/src/tree/master/usr.bin/ctags

[*2] https://www.gnu.org/software/indent