継続的インテグレーションを導入しているOSSのテスト結果の信頼性の検証

南 智孝[1]　坂口 英司[1]　伊原 彰紀[1]　松本 健一[1]

概要：継続的インテグレーション（CI）とは，ソフトウェア開発において，ビルドやテストを頻繁に繰り返し行なうことにより問題を早期に発見し，開発の効率化や納期の短縮を狙いとしており，オープンソースソフトウェア（OSS）開発でも頻繁に導入されている．本論文では，見逃し欠陥数とテスト結果の成功数に着目し，テストの見逃し欠陥数を計測することで，CI を導入している OSS では，どの程度精度良く欠陥を検出できているかを検証する．

キーワード：継続的インテグレーション，ソフトウェアテスト，OSS

1. はじめに

オープンソースソフトウェア（OSS）開発における品質保証活動は，短期間でリリースを繰り返し行い，リリース時に公開されたソースコードを不特定多数の開発者や利用者が検証，欠陥報告をし，開発者が欠陥を修正する形態が主流である．近年，OSS は民間企業や官公庁などで導入されており，高品質な OSS の需要が高まっている．しかし，OSS コミュニティが抱える課題として，テスト活動に関心のある開発者やソフトウェアテストを熟知している開発者の不足が挙げられる．このような限られたリソースでのテスト活動は困難である [1]．

OSS 開発では，テスト活動の効率化を図るために，継続的インテグレーション（CI）を導入し，テスト活動に取り組んでいる．Hilton らは，約 34,000 件の OSS プロジェクトを対象に CI の導入有無を調査した．約 4 割のプロジェクトで CI を導入していた [3]．CI とは，ソフトウェア開発において，ビルドやテストをコミット単位で繰り返し行なうことにより欠陥を早期に発見し，品質を維持できる手法である．CI に関する既存研究について，Bller らは，1108 件の CI を導入している OSS プロジェクトにおいて，少なくとも 1 回はテストが失敗しているビルドの割合を調査している．その結果，全ビルド実行回数の約 1 割でテストが失敗していた [2]．また，CI を導入している OSS プロジェクトはそうでないプロジェクトと比較して，より多くの欠陥を検出できている [6]．しかし，全ての欠陥を特定することは現実的に不可能である．Inozemtseva らは，コードカバレッジやミューテーションスコアを利用し，テストの有効性を検証することで，ソフトウェアの品質を評価しているが [4]，テストにより検出された欠陥，検出されなかった欠陥には着目していない．このような見逃し欠陥は，OSS の品質を低下させる可能性がある．我々は，見逃し欠陥数とテストの成功回数に着目し，テストの欠陥検出能力を計測することで，CI を導入している OSS プロジェクトのテスト活動を検証する．

本論文では，1000 件以上の OSS プロジェクトのビルドログのデータセット TravisTorrent [*1]，ソースコードホスティングサイト GitHub の issue tracker [*2]の欠陥報告履歴，バージョン管理システム Git [*3]の開発履歴を用いて，テストの実施によりどの程度欠陥を検出できたかを明らかにする．

2. 分析手法

2.1 見逃し欠陥の特定

図 1 は見逃し欠陥の概要図である．見逃し欠陥とは，欠陥が混入しているソースコードを対象にテストを実施し，テスト結果が成功（対象ソースコードに欠陥が混入していなかったこと）を示すことで，テストにより発見されなかった欠陥のことである．次に，見逃し欠陥の特定は以下の 5 つの手順に従って分析する．

(1) Git のコミットログから欠陥修正日時を特定する．

[1] 奈良先端科学技術大学院大学
Nara Institute of Science and Technology

[*1] https://travistorrent.testroots.org/
[*2] https://github.com/issues
[*3] https://git-scm.com/

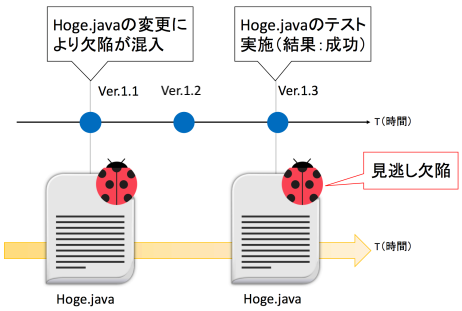

図 1　見逃し欠陥の概念図

(2) Sliwerski らが提案した SZZ アルゴリズム [5] を実装し，Git のコミットログから欠陥混入の要因となる変更とその日時を特定する．

(3) TravisTorrent のデータセットからテスト実施日時を取得する．

(4) (1)，(2) で特定した欠陥修正日時，欠陥混入の要因となる変更日時と (3) で取得したテスト実施日時を比較する．

(5) テスト実施日時が欠陥混入日時以降，かつ，欠陥修正日時以前であれば (2) で特定した変更は見逃し欠陥であると考える．

2.2　OSS プロジェクトのテスト活動の検証

図 2 は OSS プロジェクトごとのテストの欠陥検出能力の概要図である．図 2 で見逃し欠陥率の値が 1 に近い OSS プロジェクトほどテストの欠陥検出能力が低く，見逃し欠陥率の値が 0 に近い OSS プロジェクトほどテストの欠陥検出能力が高いことがわかる．本研究では，テストの欠陥検出能力を計測することで，CI を導入している OSS プロジェクトのテスト活動を検証する．テスト活動の検証に使用するメトリクスは，欠陥数，見逃し欠陥数，テストの実行回数，テスト結果の成功の数の 4 つである．OSS のテスト活動の検証は以下の 3 つの手順に従って分析を進める．

(1) 欠陥数とテストの実行回数を計測する．

(2) 欠陥数のうち，見逃し欠陥数がどれくらい存在するかを計測する．（見逃し欠陥率）

(3) テストの実行回数のうち，テスト結果が成功の数がどれくらい存在するかを算出する．（テスト成功率）

$$\text{見逃し欠陥率} = \frac{\text{見逃し欠陥数}}{\text{欠陥数}}$$

$$\text{テスト成功率} = \frac{\text{テスト結果の成功の数}}{\text{テストの実行回数}}$$

3.　おわりに

本論文は，CI を導入している OSS の積極的なテスト活動に向けて，見逃し欠陥に着目し，ソフトウェアテストの欠陥検出能力の評価手法を提案した．今後は，280 件の OSS プロジェクトを対象に見逃し欠陥数を分析し，見逃し欠陥数の多いプロジェクト，少ないプロジェクトに分類，見逃し欠陥数増減の要因分析を行うことで，OSS プロジェクトの品質向上を目指す．

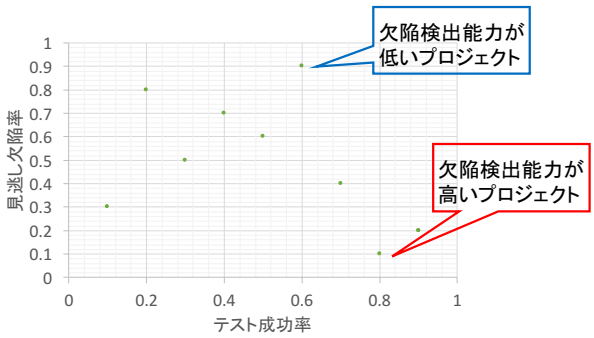

図 2　テストの欠陥検出能力の概念図

謝辞　本研究の一部は，頭脳循環を加速する戦略的国際研究ネットワーク推進プログラムによる助成を受けた．

参考文献

[1] Beller, M., Gousios, G., Panichella, A. and Zaidman, A.: When, How, and Why Developers (Do Not) Test in Their IDEs, *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, New York, NY, USA, ACM, pp. 179–190 (online), DOI: 10.1145/2786805.2786843 (2015).

[2] Beller, M., Gousios, G. and Zaidman, A.: Oops, my tests broke the build: An analysis of Travis CI builds with GitHub, *PeerJ PrePrints*, Vol. 4, p. e1984 (2016).

[3] Hilton, M., Tunnell, T., Huang, K., Marinov, D. and Dig, D.: Usage, Costs, and Benefits of Continuous Integration in Open-source Projects, *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, ASE 2016, New York, NY, USA, ACM, pp. 426–437 (online), DOI: 10.1145/2970276.2970358 (2016).

[4] Inozemtseva, L. and Holmes, R.: Coverage is Not Strongly Correlated with Test Suite Effectiveness, *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, New York, NY, USA, ACM, pp. 435–445 (online), DOI: 10.1145/2568225.2568271 (2014).

[5] Śliwerski, J., Zimmermann, T. and Zeller, A.: When Do Changes Induce Fixes?, *SIGSOFT Softw. Eng. Notes*, Vol. 30, No. 4, pp. 1–5 (online), DOI: 10.1145/1082983.1083147 (2005).

[6] Vasilescu, B., Yu, Y., Wang, H., Devanbu, P. and Filkov, V.: Quality and Productivity Outcomes Relating to Continuous Integration in GitHub, *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, New York, NY, USA, ACM, pp. 805–816 (online), DOI: 10.1145/2786805.2786850 (2015).