Vol.2015-SE-190 No.3
2015/12/15

オープンソースソフトウェアにおける テストコードの保守頻度と生存期間の分析

坂口 英司[1]　南 智孝[1]　伊原 彰紀[1]　松本 健一[1]

概要：オープンソースソフトウェア (OSS) はミッションクリティカルなシステムを始め，多くの商用システムで利活用されている．しかし，ソフトウェア開発企業が OSS 導入時に，OSS の動作確認を自ら作ることは容易ではない．OSS プロジェクトでもテストコードは作成されていることがあるが，十分に保守されていることは少ない．本論文では，OSS 開発におけるテストコードの保守活動を明らかにすることを目的に，テストコードの生存期間，テストコードの保守頻度，及び，プロダクトコードとテストコードの共進化を調査し，保守頻度と共進化が OSS 品質とどのように関係しているかを調査する．

An Analysis of Maintenance Frequency and Survival Time for Test Code in Open Source Software Project

Hideshi Sakaguchi[1]　Tomotaka Minami[1]　Akinori Ihara[1]　Ken-ichi Matsumoto[1]

1. はじめに

現在，世界中で多くのオープンソースソフトウェア (OSS) が開発されている．OSS プロジェクトのホスティングサービス Open Hub [*1] には，約 670,000 件のプロジェクトが登録され，個人だけでなく商用ソフトウェア開発企業が OSS 開発に貢献するようになり，OSS はシステム開発において必要不可欠なソフトウェアになりつつある．

しかし，企業が OSS の利用の開始を意思決定することは容易ではない．その理由として，OSS 開発の商用ソフトウェア開発とは異なる保守活動があげられる．OSS は，リリース時にソースコードが公開され，不特定多数の開発者・利用者が検証，欠陥の報告を行い，開発者が修正する．OSS 開発は，短期間でリリースを頻繁に行い，不特定多数の開発者・利用者が検証するサイクルを繰り返すことで，品質の向上を実現している [1]．そのような，品質が徐々に向上する OSS を，企業は，利用開始する前に，品質評価を行う必要がある．OSS 開発中にもテストコードが開発されており，それらは企業の品質評価にも利用することができる．

しかし，今日の多くの OSS プロジェクトはテストコードを組織的に管理できていない．Takasawa らは [2]，テストコードを公開している OSS プロジェクト 791 件を対象に，テストを実行した．その結果，ビルドが成功したプロジェクトは 452 件 (57.1%) であった．また，ビルドに成功したプロジェクト 452 件を対象にテストの実行した結果，全てのテストケースが成功したプロジェクトは 117 件 (14.8%) であり，テストコードが存在していても，実行できるテストコードは数少ない．この理由として Andy らは，ソフトウェアを構成するソースコード（プロダクトコード）とテストコードの共進化を挙げている．ソースコードの共進化とは，プロダクトコードが作成・変更されたタイミングで，テストコードも作成・変更することである．一般に，欠陥の発見は早ければ早いほど修正のコストが少なくなることが知られており，ソースコードの共進化は欠陥の早期発見に寄与する手法であると考えられる．しかし，OSS 開発においては各開発者が自由に開発を行うため，ソースコードの共進化が起こりにくいと考えられる．

本論文では，OSS 開発におけるテストコードの保守活動を明らかにすることを目的に，テストコードの生存期間，

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

[*1] https://www.openhub.net/

テストコードの保守頻度，及び，プロダクトコードとテストコードの共進化を調査し，保守頻度と共進化が OSS 品質とどのように関係しているかを調査する．具体的には，Apache HTTP Client プロジェクトを対象に，テストコードの生存期間，保守頻度，プロダクトコードとテストコードの共進化の有無，プロダクトコードの品質を計測する．

生存期間:テストコードが作成されてから，削除されるまでの日数．削除されていない場合は，現時点までの日数
保守頻度:テストコードの生存期間中の変更回数
共進化:プロダクトコード変更と同時にテストコードが変更された割合
プロダクトコードの品質:プロダクトコードへの欠陥混入数

本論文の貢献は，OSS 開発におけるテストコードの管理状況とプロダクトコードの欠陥混入数との関係を明らかにすることで，OSS 開発でのテストコードの管理手法に関する知見を得られることである．

続く 2 章では，関連研究を述べ本論文の立場を明らかにする．3 章では，ケーススタディの概要とその結果を述べる．4 章では，ケーススタディについての考察を行い，最後に 5 章で本論文のまとめと今後の課題を述べる．

2. 関連研究

近年，OSS 開発におけるテストの現状を調査する研究が増えている．Pavneet ら [3][4] は，ソースコードホスティングサイト GitHub[5] から 50,000 件の OSS を取得し，テストケースの有無を調査した．過半数のプロジェクトはテストケースを持っておらず，また，Android アプリケーションのうち，テストカバレッジが 40%を超えるプロジェクトは全体の 9%程度であることがわかった．このほか，Tosi ら [6] は，33 件の OSS から実践されているテスト手法や，テストのドキュメントの有無などを調査している．67%のプロジェクトではテストのドキュメントを保持しておらず，テストのドキュメントが存在していても，不完全であったり古い情報しか掲載されていない．また，Takasawa らは [2]，テストコードを公開している OSS プロジェクト 791 件を対象に，テストを実行した．その結果，ビルドが成功したプロジェクトは 452 件 (57.1%) であり，その 452 件を対象にテストの実行した結果，全てのテストケースが成功したプロジェクトは 117 件 (14.8%) であった．これらの調査から，多くの OSS プロジェクトでは必ずしもテストが十分に管理されているとは言いがたい．しかし，全てのプロジェクトでテストが存在していないわけではなく，多くの調査では，ある任意の時点において，テストコードの有無を調査している．

昨今では，テストコードの存在，テストが実行できなくなった原因などを調査するためのツールが提案されている．Pinto らは，テストの保守の簡便化を目的として，TestEvol というテストの変更を可視化するツールを提案した [7]．また，Andy らは [8]，プロダクトコードとテストコードの共進化をグラフ化するツールを開発した．開発したツールにより，プロダクトコードとテストコードが共進化していないことが多く，共進化していないことが原因で，ビルドできないテストコードが存在していると考えられる．本論文では，リリース時にテストコードが実行できなくなる理由として，プロダクトコードとテストコードの共進化とテストコードがビルドできる期間（生存期間）を調査する．

表 1　Apache HTTP Client のプロダクトコードとテストコード

	件数
プロダクトコード	1403
テストコード	398
テストコード 1 件に対する検証対象プロダクトコード（平均）	6.77

3. ケーススタディ

本章では，Apache HTTP Client プロジェクトを対象に，テストコードの生存期間，テストコードの保守頻度，及び，プロダクトコードとテストコードの共進化の度合いのそれぞれの分布を調査した上で，最後に，各テストコードの調査とプロダクトコードへの欠陥混入数の関係を調査する．テストコードが削除された一要因を理解するために，我々が調査時点で削除されていたテストコードと現在でも存在しているテストコードを区別して，生存期間，保守頻度，共進化の度合いの分布を調査する．

3.1 データセット

Apache HTTP Client [*2] は，Apache ソフトウェア財団の傘下にある再利用可能な Java コンポーネントをまとめた Apache のプロジェクトの一つで，HTTP クライアントアプリケーションの構築をサポートするクラスライブラリである．Apache HTTP Client は，Github でソースコードをバージョン管理している [*3]．

本論文では，Github で管理される java ファイルを分析対象とした．今回は，各コンポーネントの/src/test/ディレクトリ下に保管されいるソースコードをテストコードとして扱い，それ以外のソースコードをプロダクトコードして扱った．また，今回は，テストコードの命名規則及びテストコードでインポートしているプロダクトコードをテスト対象コードとして扱い，リンキングを行った．

3.2 テストコードの生存期間

[アプローチ] ソフトウェアの動作確認をするためにテストコードが開発されるが，時間の経過とともに，変更されなくなる事例が見られる [8]．テストコードが使用されなくなって削除されることもある． 本論文では，生存期間を，

[*2] http://hc.apache.org/httpclient-3.x/
[*3] https://github.com/apache/httpclient/

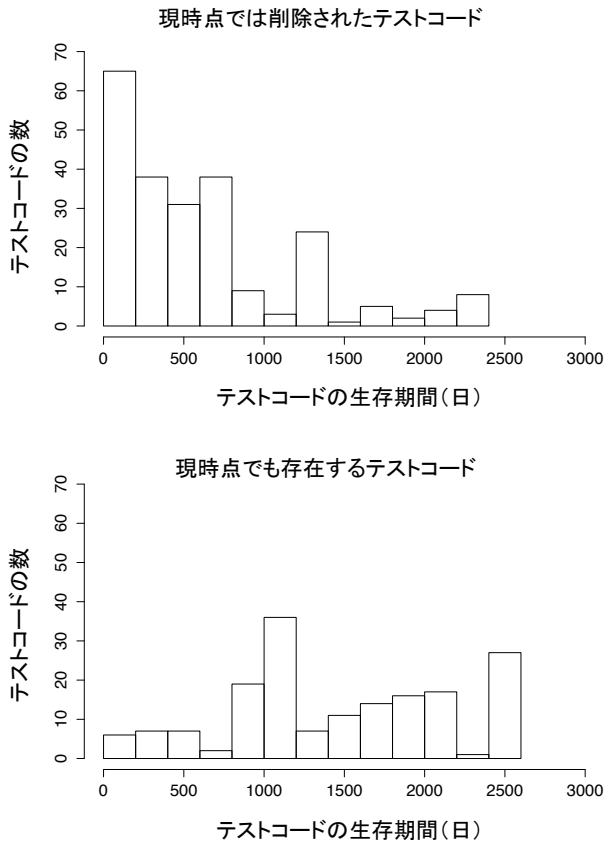

図 1　テストコードの生存期間

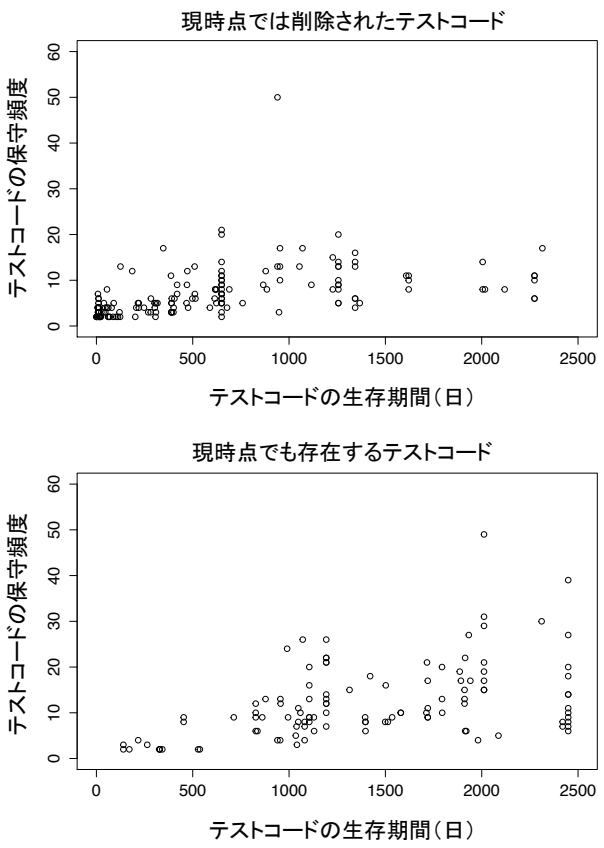

図 2　テストコードの保守頻度

テストコードが作成されてから削除されるまでの期間を計測する．テストコードが削除されていない場合は，テストコードが作成されてから現時点までの期間として計測する．計測した生存期間は，保守頻度，共進化の度合いの分布を調査するときに用いる．

[結果] 図 1 は，テストコードの生存期間を表したヒストグラムである．上図は現時点では削除されたテストコード，下図は現時点でも存在するテストコードを対象とした図である．両テストコードとも，5 年以上リポジトリに存在していたテストコードが存在しているが，現時点で削除されたテストコードは，1 年以内に削除されたテストコードが 50%以上を占め，多くのテストコードは開発者にとって必要のないテストコードと判断されたことが示唆される．

3.3　テストコードの保守頻度

[アプローチ] プロダクトコードが変更された時点でテストコードも変更される可能性が高いが，テスト内容，テスト範囲を変更する場合は，テストコードのみの変更も考えられる．本論文では，保守頻度として，テストコードがリポジトリに存在する期間中の変更頻度を計測する．開発者がテストがどのように必要としなくなるのかを理解するために，テストの生存期間が長期化するにつれてテストコードの保守頻度を分析する．

[結果] 図 2 は，テストコードの保守頻度を表した散布図である．上図は現時点では削除されたテストコード，下図は現時点でも存在するテストコードを対象とした図である．スピアマンの順位相関係数は，削除されたテストコードが 0.69，現時点で存在するテストコードは 0.48 であり，どちらも生存期間が長くなるにつれて保守頻度が多くなる正の相関がある，削除されたテストコードの保守頻度は，15 回未満に分布している．その一方で，現時点でも存在するテストコードは，それ以上変更されており，特に生存期間が長いテストコードへの保守頻度は，削除されたテストコードに比べて多いため，開発者が使用するテストコードと使用しないテストコードは区別できる可能性がある．

3.4　プロダクトコードとテストコードの共進化の度合い

[アプローチ] 1 章でも述べたように，本論文が対象とする共進化は，プロダクトコードが作成・変更されたタイミングで，テストコードも作成・変更することである．本論文では，テストコードの保守頻度と同様に，開発者がテストがどのように必要としなくなるのか，また，プロダクトコードとどの程度，共進化すべきなのかを明らかにする．共進化の度合いは，以下の式によって算出する:

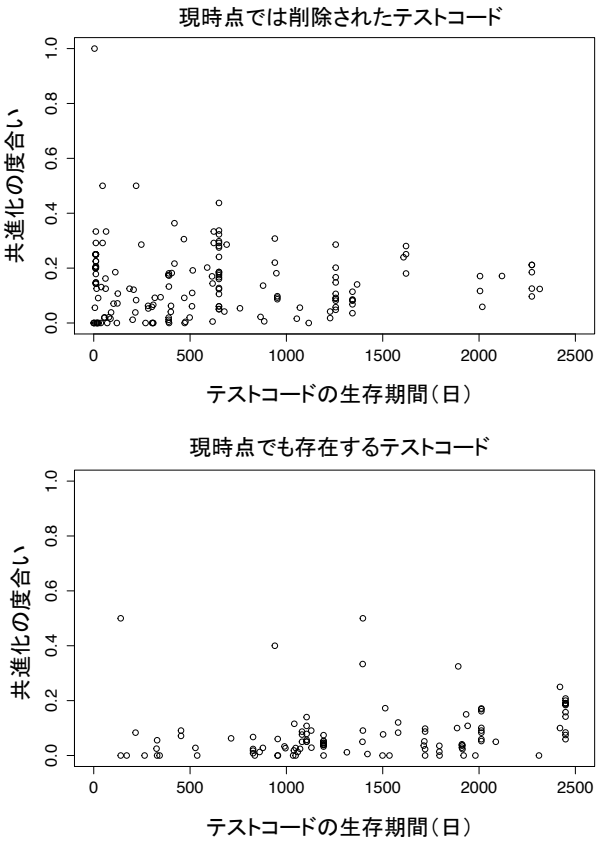

図 3　プロダクトコードとテストコードの共進化の度合い

$$\text{共進化の度合い} = \frac{\text{プロダクトコードとテストコードが同じコミットで変更された回数}}{\text{プロダクトコードとテストコードの合計コミット回数}}$$

[結果] 図 3 は，プロダクトとテストコードの共進化の度合いを表した散布図である．上図は現時点では削除されたテストコード，下図は現時点でも存在するテストコードを対象とした図である．スピアマンの順位相関係数は，削除されたテストコードが 0.05，現時点で存在するテストコードは 0.42 であり， 削除されたテストコードは生存期間と共進化の度合いにはほとんど相関がないが，現時点で存在するテストコードは正の相関が見られる．削除されたテストコードは，相関は見られないものの，テストコードが作成された時点では現在も存在するテストコードに比べて高い確率で共進化しているが，生存期間が長期化するにつれて低下する．その一方で，現時点でも存在するテストコードは生存期間が長いものほど共進化している．開発者は一時的に使用するテストコードと長期的に使用されるテストコードが存在することが示唆される．

3.5　テストコード保守管理下におけるプロダクトコードの品質

[アプローチ] テストコードの生存期間，保守間隔，共進化の度合いがプロダクトコードの品質との関係を分析するために，それぞれの期間，頻度，度合いを中央値で二分し，テストコードの管理度合いを 8 つのケースに分類する．各分類は表 2 に表す．そして，各テストコードが評価するプロダクトコードへの欠陥混入数を比較する．

表 2　テストコードのメンテナンスパターンとプロダクトコードへの欠陥混入数との関係

生存期間	保守頻度	共進化	一ヶ月あたりの平均欠陥混入数	テストコードの数
長い	低い	する	0.047	24
長い	低い	しない	0.069	18
長い	高い	する	0.112	42
長い	高い	しない	0.056	54
短い	低い	する	5.941	61
短い	低い	しない	0.296	54
短い	高い	する	0.327	14
短い	高い	しない	0.109	9

[結果] 表 2 は，各ケースの欠陥混入数を表している．生存期間の中央値は 845.5 日，保守頻度の中央値は 8 日，共進化度合いの中央値は 0.83 日だった．生存期間（短い），保守頻度（低い）というケース，または，生存期間（長い），保守頻度（高い）というケースに該当するテストコードが多く見られた．特に生存期間（短い），保守期間（低い），共進化（する）というケースに欠陥混入数が多く，たとえ共進化していても十分なテストが実施されていない，一時的なテストしかされていない場合は欠陥が混入されやすいことが示唆される．

4.　議論

4.1　テストコードの必要性

ケーススタディでは，Apache HTTPClient プロジェクトを対象として，テストコードの生存期間，保守頻度，共進化の度合いを調査し，最後に，プロダクトコードの品質との関係について調査した．

削除されるテストコードは，生存期間が短く，保守頻度が少ない一方で，共進化の度合いは高かった．一時的に作られたテストコードは，プロダクトコードと同時変更が必要であった場合が多い．このようなテストコードは，OSS 開発のように開発者の参加/離脱が繰り返されるプロジェクトでは，継続的に保守することが難しく，時間経過とともに実行できなくなり削除されたテストコードであることが考えられる．特に生存期間（短い），保守頻度（低い），共進化（する）に該当するテストコードが検証するプロダクトには欠陥が混入している可能性が高く，当該プロダクトコードは利用者側で動作確認等のテストを実施する必要があると考える．

また，現在も存在するテストコードは生存期間が長く，保守頻度が多い一方で，共進化の度合いは低かった．長期的に使用されるテストコードは，プロダクトコードと共進

化せずとも，定期的に変更され，継続的に保守されていることから生存期間も長いことが示唆され，当該テストコードが検証するプロダクトコードは OSS プロジェクトで定期的に保守されているため，利用者は改めて検証するコストを下げることができるかもしれない．

4.2 制約

本論文は，Apache HTTP Client プロジェクトのみを対象として調査しており，結果の妥当性が十分ではない．また，テストコードはバージョン管理システムに存在していても，実際に使われていたかは確認できていない．Mozilla Firefox プロジェクトなどでは，テストコードの実行結果をツール MozTrap で管理しているため，このようなツールを使ってるプロジェクトを対象に分析すると実行可否についての分析が可能になると考える．本論文では，テストコードを全て同一に扱っているが，検証目的はテストコードによって異なると考えられる．今後は，テストコードの目的に合わせて，保守頻度等の違いを調査する．

5. おわりに

本研究では，Apache HTTP Client プロジェクトを対象に，OSS 開発におけるテストコードの保守活動を明らかにすることを目的に，テストコードの生存期間，テストコードの保守頻度，及び，プロダクトコードとテストコードの共進化を調査し，保守頻度と共進化が OSS 品質とどのように関係しているかを調査した．

削除されるテストコードは，生存期間が短く，保守頻度が少ない一方で，共進化の度合いは高かった．特に，生存期間（短い），保守頻度（低い），共進化（する）に該当するテストコードが検証するプロダクトには欠陥が混入している可能性が高いことがわかった．一方で，現在も存在するテストコードは生存期間が長く，保守頻度が多い一方で，共進化の度合いは低いことがわかった．

OSS 開発では，一時的に作られたテストコードと長期的に使用されているテストコードで更新頻度や共進化の度合いは異なり，利用者にとって利活用できるテストコードとできないテストコードを判別可能であることが示唆された．今後は，テストコードの内容によって，開発者の保守作業の違いを明らかにしていく予定である．

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] Raymond, E. S.: *The cathedral and the bazzar: musings on linux and open source by an accidental revolutionary*, O'Reilly and Associates, Sebastopol, CA (1999).
[2] Takasawa, R., Sakamoto, K., Ihara, A., Washizaki, H. and Fukuzawa, Y.: Do open source software projects conduct tests enough?, *Proc. of the 15th International Conference of Product Focused Software Development and Process Improvement (PROFES'14)*, pp. 322–325 (2014).
[3] Pavneet Singh, K., Tegawende F., B., Lo, D. and Lingxiao, J.: Adoption of Software Testing in Open Source Projects–A Preliminary Study on 50,000 Projects, *Software Maintenance and Reengineering (CSMR), 2013 17th European Conference on*, Genova, Italy, IEEE, pp. 353–356 (2013).
[4] Kochhar, P., Thung, F., Nagappan, N., Zimmermann, T. and Lo, D.: Understanding the Test Automation Culture of App Developers, *Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on*, Graz, Austria, IEEE, pp. 1–10 (2015).
[5] GitHub: GitHub, GitHub (online), available from ⟨https://github.com/⟩ (accessed 2015-11-10).
[6] Davide, T. and Abbas, T.: A Survey on How Well-Known Open Source Software Projects Are Tested, *Software and Data Technologies*, Athens, Greece, Springer Berlin Heidelberg, pp. 42–57 (2010).
[7] Pinto, L., Sinha, S. and Orso, A.: TestEvol: A tool for analyzing test-suite evolution, *Software Engineering (ICSE), 2013 35th International Conference on*, California, USA, IEEE, pp. 1303–1306 (2013).
[8] Andy, Z., Bart, Van, R., Arie, van, D. and Serge, D.: Studying the co-evolution of production and test code in open source and industrial developer test processes through repository mining, *Empirical Software Engineering*, Vol. 16, pp. 325–364 (2010).