ソフトウェア開発において 継続的インテグレーションが効果を発揮する条件

今井菜緒子[1]　鷲崎弘宜[1]　津田直彦[1]　深澤良彰[1]

概要：CI（Continuous Integration）とは，ソフトウェア開発において，コードのビルドやテストを自動で行うツールである．CI 利用のメリットについては報告があるが，どのような開発プロジェクトで CI の効果が発揮されやすいかについては明らかになっていない．本調査では，開発プロジェクトの CI の利用有無と課題解決期間に着目し，その条件について調査を行った．その結果，チームサイズが大きいプロジェクトで CI の効果が発揮されることを確認した．

1. はじめに

CI（Continuous Integration）とは，ソフトウェア開発において，コードのビルドやテストを自動で行うツールである．CI を導入することで，Pull Request の処理数の増加や，レビューコメントの節約など，ソフトウェア開発の円滑化が促進される[1][2].

先行研究[3]では，課題解決期間を評価軸として，どのような開発プロジェクトで CI の効果が発揮されやすいかについて調査を行っていた．調査では，CI を利用している開発プロジェクトにおいて，コード変更量とチームサイズは課題解決期間に影響を与えないと報告されている．しかし，CI を利用していないプロジェクトとの比較を含めた調査は行っていない．

本稿では，CI の利用有無にも着目し，CI の効果が発揮されやすい開発プロジェクトの特徴について報告する．本調査により，CI を導入する指標を明らかにすることで，ソフトウェア開発プロジェクトへの貢献を期待する．

2. 関連研究

2.1 CI の利用

CI ツールを導入するプロジェクトの中には，開発を進めるうちに使用規則が守られなくなり，CI の効力が弱まるものもある[4]．また，CI を利用するためには，複雑なビルドログを理解する必要がある[5]．このようなデメリットがあることから，CI ツールの乗り換えや放棄に関する調査を行う研究もある[6]．本調査で扱うプロジェクトは，Travis CI の利用を基本とし，Travis CI を利用しているプロジェクトと，Travis CI を放棄したプロジェクトに分類される．

2.2 Issue 解決者

ソフトウェア開発には様々な役割があるが，「Issue の解決者」は定義されていない[7]．Issue が閉じられた時刻を Issue が解決した時刻として扱う研究[8]があるため，本調査では，Issue Closer を Issue 解決者と仮定した．また，Issue Reporter の多くは Issue 解決者でもあるという報告[9]があるため，Issue Reporter と Issue Closer の関係についても調査を行った．

3. 調査

3.1 研究課題

本調査の目標は，CI がどのような開発プロジェクトで効果を発揮しやすいかを明らかにすることである．ここで，CI が効果を発揮するとは，課題解決期間が短くなることと定義する．また，Issue 解決者は Issue Closer と仮定する．

本調査では，先行研究での着眼点に加え，CI の利用有無や，Issue Reporter と Issue 解決者の関係にも着目し，以下の課題を定めた．

RQ：CI の利用有無とチームサイズは課題解決期間に影響を与えるか？

回答のために，CI を継続利用しているプロジェクトと利用を放棄したプロジェクトをチームサイズの大小で分け，課題解決期間の分布を確認する．また，得られた結果が Issue Reporter と Issue 解決者が同じであることが関係するかについても調査を行う．

3.2 調査の全体像

本調査は，Travis Torrent で公開されている開発プロジェクトのデータの中から，欠損のないリポジトリ 1,183 個を対象に行った．その内，CI を継続利用しているリポジトリは 893 個，CI 利用を放棄したリポジトリは 290 個であった．データの中からプロジェクト名とチームサイズを取得し，GithubAPI を使用して課題解決期間と Issue Reporter，Issue Closer の情報を取得した．

まず，CI の利用有無とチームサイズの大小を分けて課題解決期間の分布を見て，CI とチームサイズが課題解決期間に与える影響について調査する．次に，取得できた Issue の数に対して Issue Reporter と Issue Closer が同じだった割合を調べ，CI の利用有無やチームサイズの大小との関係について調査する．

3.3 データセット

データセットとして，先行研究同様，Travis Torrent で公開されているリポジトリを使う[10][11]．Travis Torrent データセットは，2015 年 8 月時点で Travis CI を使用していた 1,183 個のアクティブな OSS リポジトリに関するデータや，Travis CI のビルドログが公開されている．

Travis CI を継続利用しているか否かは，リポジトリ内の

1 早稲田大学 Waseda University

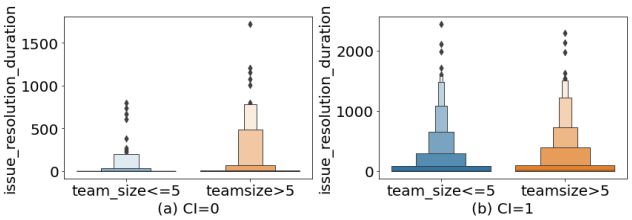

図 1　課題解決期間の差

Travis CI 設定ファイルの有無で判断する．設定ファイルが確認できない場合，Travis CI の利用を放棄したものとする．Issue Reporter, Issue Closer は，Close された Issue の最新 100 件をサンプリングした．課題解決期間は，Close された Issue 100 件分の open から close までの期間の中央値とした．

3.4 結果と考察

CI の利用有無とチームサイズの大小で分けた課題解決期間の箱ひげ図を図 1 に示す．CI=1 は CI を利用していることを，CI=0 は CI を利用していないことを示す．チームサイズの大小は，今回扱ったデータの中央値である 5 を基準とし，チームサイズが 5 以下であれば小さなチーム，5 より大きければ大きなチームとしている．小さなチームと大きなチームに対し，優位水準を 0.05 としてマンホイットニーの U 検定を行ったところ，帰無仮説は棄却されたため，小さなチームと大きなチームの課題解決期間には差があることが分かった．両群の中央値は等しかったが，平均値はチームサイズの大きなチームの方が大きかった．また，CI を利用している小さなチームと大きなチーム，CI を利用していない小さなチームと大きなチームのそれぞれに対し，優位水準を 0.05 としてマンホイットニーの U 検定を行ったところ，どちらも帰無仮説は採択されたため，CI の利用有無にかかわらず，大きなチームと小さなチームの課題解決期間には差が無いことが分かった．

図 1 より，CI を利用していない場合，チームサイズが大きいと課題解決期間が長くなる傾向があると分かる．対して，CI を利用している場合，チームサイズが大きくとも課題解決期間に大きな差は見られなかった．以上より，チームサイズが大きい場合，CI がよく効果を発揮すると考えた．

検定では，CI を利用していない小さなチームと大きなチームに有意な差は見られなかったが，図により課題解決期間の分布に差があることを確認した．そのため，課題解決期間の分布に差がある理由を特定するための追加調査を行った．CI を利用していない小さなチームでは，課題解決期間が短くなる傾向が見られたが，これは，少人数チームの場合，Issue Reporter が集中して課題を解決し，Issue Reporter が Issue 解決者にもなる傾向があるのではないかと考えた．そこで，CI を継続利用しているプロジェクトと利用を放棄したプロジェクトをチームサイズの大小で分け，Issue Reporter と Issue Closer が同じ Issue の割合を確認した．ここで，Issue 解決者は Issue Closer であると仮定している．しかし，各分類において得られた割合の分散，平均値，中央値に有意な差は見られなかった．この結果から，Issue Reporter と Issue 解決者が同じであることは，課題解決期間に影響を与えないと考えられる．

3.5 妥当性への脅威

本調査では，Travis Torrent からデータを取得しているため，対象としている CI は Travis CI のみであることに注意する必要がある．CI を利用していないプロジェクトは Travis CI を放棄したとしているが，実際には別の CI へ乗り換えていることもある．

また，Travis Torrent から得られたデータと Issue に関するデータの取得時刻は異なるため，データを再取得すると結果が変化する可能性がある．

4. おわりに

本研究では，CI が効果を発揮するプロジェクトの条件について調査を行った．CI は，チームサイズが大きいプロジェクトで効果を発揮することが分かり，それをもたらす理由の候補として検討した Issue Reporter と Issue Closer の関係については，特別な傾向が見受けられないことを確認した．Issue Reporter と Issue 解決者の関係を調査するには，Issue 解決者の定義を見直す必要がある．

参考文献

[1] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu and V. Filkov, “Quality and Productivity Outcomes Relating to Continuous Integration in GitHub”, ESEC/FSE 2015, 2015, pp. 805-816.
[2] N. Cassee, B. Vasilescu and A. Serebrenik, “The Silent Helper: The Impact of Continuous Integration on Code Reviews,” SANER 2020, 2020, pp. 423-434.
[3] 飯島楓，新井珠旺，津田直彦，鷲崎弘宜，深澤良彰，“オープンソース開発における継続的インテグレーションの効果を発揮する条件”，SES 2020，2020，ポスター
[4] C. Vassallo, S. Proksch, H. C. Gall and M. D. Penta, “Automated reporting of anti-patterns and decay in continuous integration,” ICSE '19, 2019, pp. 105-115.
[5] C. Vassallo, “Enabling Continuous Improvement of a Continuous Integration Process,” ASE 2019, 2019, pp. 1246-1249.
[6] D. G. Widder, M. Hilton, C. Kästner and B. Vasilescu, “A conceptual replication of continuous integration pain points in the context of Travis CI,” ESEC/FSE 2019, 2019, pp. 647–658.
[7] Z. Wang and D. E. Perry, “Role Distribution and Transformation in Open Source Software Project Teams,” APSEC 2015, 2015, pp. 119-126.
[8] J. Anvik, L. Hiew and G. C. Murphy, “Who should fix this bug?,” ICSE '06, 2006, pp. 361-370.
[9] T. F. Bissyandé, D. Lo, L. Jiang, L. Réveillére, J. Klein and Y. L. Traon, “Got issues? Who cares about it? A large scale investigation of issue trackers from GitHub,” ISSRE 2013, 2013, pp. 188-197.
[10] “Travis Torrent”, https://travistorrent.testroots.org/
[11] M. Beller, G. Gousios and A. Zaidman, “TravisTorrent: Synthesizing Travis CI and GitHub for Full-Stack Research on Continuous Integration,” MSR 2017, 2017, pp. 447-450.