コーディング活動ログ収集フレームワークの構築と演習活動がコーディングプラクティスに与える影響の実験的分析

Development of Coding Activity Log Collecting Platform and Experimental Activity Analysis for Learners' Coding Practice

野口靖浩[*1]，増山寧浩[*1]，小暮悟[*1]，山本頼弥[*2]，山下浩一[*2]，小西達裕[*1]
Yasuhiro NOGUCHI [*1], Yasuhiro MASHIYAMA[*1], Satoru KOGURE [*1],
Raiya YAMAMOTO[*2], Koichi YAMASHITA[*2], Tatsuhiro KONISH [*1],
[*1] 静岡大学情報学部
[*1]Faculty of Informatics, Shizuoka University
[*2] 常葉大学経営学部
[*2]Faculty of Business Administration, Tokoha University
Email: noguchi@inf.shizuoka.ac.jp

あらまし：近年，ソフトウェア開発の現場では自動テストが用いられており，自動テストに関する知識及びスキルが求められている．自動テストをプログラミング学習に採用する事例はあるが，演習の中で実際に学習者が自動テストをどのように活用しており，その経験が学習者のコーディングプラクティスにどのような影響を及ぼすのか，については十分には明らかにはされていない．本稿では，長期のプログラミング演習を対象として学習者のコーディング活動ログを収集するフレームワークを開発し，そこで収集した活動ログを分析することで演習活動がコーディングプラクティスに与える影響の分析を試みる．
キーワード：プログラミング，学習活動分析，コーディングスタイル，自動テスト，TDD

1. はじめに

近年，ソフトウェア開発現場の多くで自動テストが用いられており(1)，自動テストに関する知識及び実践的なスキルが求められている．しかしながら大学などで行われる初期のプログラミング学習において自動テストが採用されることは多くない．この原因のひとつとして，テストケースの作成スキルやテスト結果をデバッグに活用する能力が必要であり，初学者に要求することが難しいことが挙げられる．そのような現状に対してテストコード自動生成による実施支援(2)(3)や，テスト駆動開発のコーディング実施支援(4)の研究が行われている．大橋らの研究(2)ではテストコードの自動生成支援により，初学者でも自動テストを作成して演習を行えることを確認している．しかしながら，このテストがどのように役立てられコードの完成に寄与したのか，また，学習者が自動テストを含むコーディング方法を修得し実施するようになったのか，などは明らかにされていない．

そこで本研究では，学習者のコーディング活動を長期間収集し，学習者が自動テストを用いた演習を行う中で，自分のコーディングプラクティスに自動テストをどの程度取り入れられるのかどうか，コーディングプラクティスと演習効率との間にどの程度の関係があるのか，を調査・分析する．

2. 関連研究

Shynkarenko ら(5)は統合開発環境（IDE）上でのデバッグプロセスを対象とした分析を行い，プログラマと IDE のインタラクションを分析することはデバッグプロセスを効率的に分析し，研究者が開発者の行動を解釈するのに役立つと指摘している．Zhevaho ら(6)は IDE とプログラマのインタラクションを追跡するツールの現状を調査し，それらが主に細かいタイピングログを使用してプログラミングセッションを再生できるタイムライン機能をもとに設計されていることと，それらの分析ではプログラマはブレークポイントよりも printf デバッグのようなより単純なデバッグ手法を好む傾向が示されていることを明らかにした．本稿はプログラマのこのような基本的な傾向に対して，自動テストに基づく演習活動が学習者のコーディングプラクティスにどの程度の影響を与えるのかを分析しようとするものである．

3. データ収集

情報系の学部で開講される実験科目に自動テストを導入しコーディング活動履歴を収集した．毎回課題の変わる演習では，学習者が作成した自動テスト及びそれを実施することによる利点を感じにくいことから，14 回（各回 3 コマ: 270 分）に渡ってソフトウェアの機能を加法的に開発する演習を対象とした．内 7 回が必須課題，残り 7 回は必須課題を完了した者のみを対象とする追加課題である（完了していない者は必須課題を進める）．自動テストによる開発の経験のない 113 名を対象とした．本演習の開発課題は簡易的な購入言語のコンパイラであり，最終的には演習用の CPU 上で実行可能な機械語を出力までサポートする．最終的な開発規模は約 80 クラス，4000 行である．開発対象のモジュールのテスト容易性の設計不備による行き詰まりを避けるため，毎回基本となるテストコードの雛形とテストケースの一

例を与えることとした．従って，学習者はテストケースの設計と実装並びに，作成した自動テストを実行しながらプロダクトコードを開発することに集中することができる．

コーディング活動ログを収集するために図 1 に示す VSCode の拡張機能を開発した．収集対象のデータは，周期的ないしはイベントごとの，コード，編集履歴，開いているファイルやカーソル等の位置情報，ビルド・実行の結果，テスト実行対象とテスト結果である．コーディング活動ログは各回課題完成時にまとめて回収することとした（表 1）．

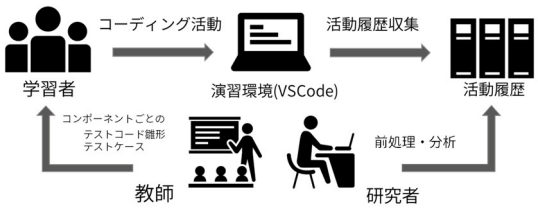

図 1 コーディング活動ログ収集フレームワーク

表 1：収集コーディング活動ログ概要[1]

課題	第 1 回	第 2 回	…	第 6 回	第 7 回
データ収集人数（人）	100	93	…	65	49
テスト実行回数平均（回）	53.3	122	…	93.4	143
演習時間中央値（分）	85.5	208.6	…	102	225.3
演習時間の標準偏差	71.3	97	…	75.7	76.3

4. 分析と考察

収集したコーディング活動ログの可視化による各学習者のコーディング活動の観察と，自動テストを伴う演習活動に関する指標値を設定し，前半と後半の演習課題のコーディング活動の差の分析を行った．

図 2 にコーディング活動ログの可視化例を示す．横軸が演習開始時からの時間経過（VSCode の操作を伴う時間帯のみ），縦軸が参照・編集・ビルド・実行・テスト活動をまとめたものと，各参照・編集ファイルを示す．この学習者・演習の活動ログでは，演習時間の後半から，コードの編集の中で周期的に関数単位のテスト（3 行目・4 行目）を実行していることが読み取れ，特に後半の演習課題において定期的なテスト実行が行われている様子が観察された．

コーディングプラクティスに自動テストが浸透したかの指標として，編集後にテストを実行するまでの平均時間が短くなったか（リグレッションテストが行われているか），テスト失敗から成功までの平均時間が短くなったか（失敗テストを放置しないようになったか）を設定した．第 1 回は演習及び環境に慣れる必要があるので，第 2 回と第 6,7 回の課題分を比較することとした．表 2 に平均時間の差をウィルコクソンの順位和検定を用いた結果を示す．後半の課題では，コーディング活動の中で自動テストが活用されている可能性が示されたが，課題による影響も大きいことが想定される．

[1] 活動ログは課題ごとに集計している．例えば第 7 回のログは，第 7 回の実施日のログではなく，第 7 回課題に取り組み始めてから完成させるまでのログを示す．

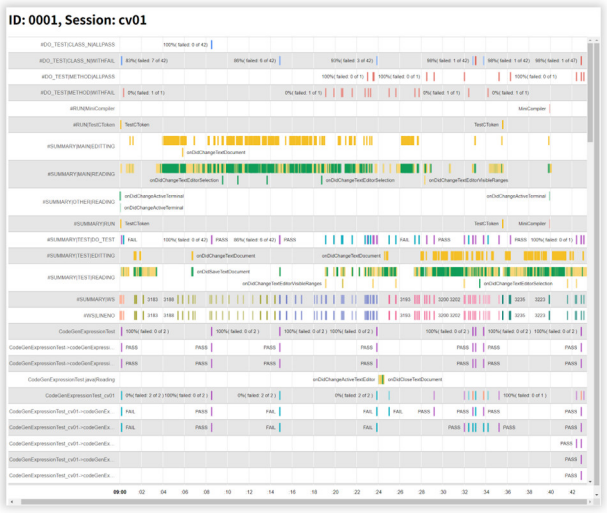

図 2 コーディング活動ログの可視化例

表 2：前半と後半の演習課題の活動の差

		Z	p	効果量
編集後のテスト実行までの平均時間	第 2 回>第 6 回	1397	<.001**	0.633
	第 2 回>第 7 回	631	0.164	0.1674
テスト失敗から成功までの平均時間	第 2 回>第 6 回	1247	0.024*	0.334
	第 2 回>第 7 回	721	<.001**	0.458

5. まとめ

ソフトウェア課題を加法的に開発するプログラミング演習の活動ログを収集し，長期の演習活動がコーディングプラクティスに与える影響の分析を試みた．後半の課題にかけて，自動テストの活用がコーディングプラクティスに取り入れられる傾向が観測された一方で，学習者や課題の差異も大きく，更に精緻な分析を行う必要性が示された．

参考文献

(1) PractiTest: “The State of Testing Report 2022”, https://www.practitest.com/assets/pdf/state-of-testing-report-2022v10.pdf, (2023/10/22 参照)

(2) 大橋旭雄，神長裕明，中村勝一，森本康彦，宮寺康造: “ソフトウェア開発演習における自動テスト実施支援システム”，電子情報通信学会教育工学研究会技術研究報告, Vol.114, No.513, pp.13-18 (2015)

(3) 鎌田高如，樋口昌宏: “Web アプリケーションを対象とした高網羅率の単体テスト自動生成について”，情報科学技術フォーラム講演論文集，Vol.10，No.1，pp.245-246 (2011)

(4) Nobuo F., Yukiko M., Toru N., Kan W., and Noriki A: “A Java Programming Learning Assistant System Using Test-Driven Development Method”, IAENG International Journal of Computer Science, Vol.40, pp.38-46 (2013)

(5) V. I. Shinkarenko, Oleksandr Zhevaho: “Development of a toolkit for analyzing software debugging processes using the constructive approach”, Eastern-European Journal of Enterprise Technologies, Vol.5, No.2, pp.29-38 (2020)

(6) Oleksandr Zhevaho: “An Overview of Tools for Collecting Data on Software Development and Debugging Processes from Integrated Development Environments”, Science and Transport Progress, No.3, pp.24-37 (2021)