DEIM Forum 2020 I1-3(day1 p56)

プログラミング演習課題の過程評価のための評価指標の検討

谷口　哲朗† 　前川　絵吏† 　三嶋　哲也† 　清光　英成† 　西田　健志†

† 神戸大学大学院国際文化学研究科　〒 657–8501 神戸市灘区鶴甲 1 － 2 － 1

E-mail: †{tetsuro.taniguchi,eri.maekawa,tetsuya.mishima}@mulabo.org, ††kiyomitu@kobe-u.ac.jp, †††tnishida@people.kobe-u.ac.jp

あらまし　2020 年にプログラミング教育が小学校で必修化されるなど, 教育現場での教材としてプログラミングが用いられることが今後増えていくと予想される．大学ではプログラミング演習などの，受講生にプログラムを書かせてその内容を評価する授業が行われている．このような授業において学習者の現状を知るためにはプログラミング開発過程でのコードを精読あるいは他の受講生と比較が有効である．プログラミング開発環境の平易化により幅広い学習者がプログラミングを学ぶようになってきた．それに伴い教師が受け持つ受講生数が増えつつあり，学習者が困難に直面した時点での支援を現状と同等にきめ細かく行うことが困難となりつつある．そこで本稿ではプログラミングの授業におけるコーディングの試行回数・更新頻度・実際の成績評価などのメタデータを統計的に分析することで，コードを精読しなくても受講生のコーディング過程を評価できる可能性について議論する．

キーワード　プログラミング教育, 情報要約

1 はじめに

近年プログラミング教育が盛り上がりを見せている．文部科学省は 2020 年に小学校におけるプログラミング教育を必修化するとしており [1]，その後も中学校・高校においてプログラミングが必修化されることになっている．大学においてはプログラミング演習などの，受講生にプログラムを書かせてその内容を評価する授業が多く行われている．このように教育現場での教材としてプログラミングが用いられることが今後ますます増えていくと予想される．

また，プログラミング環境の平易化・教育現場におけるコンピュータ設備の拡充により幅広い学習者がプログラミングを学ぶことが可能となっている．プログラミング演習などの授業において学習者の現状を知るためにはプログラミング開発過程でのコードを精読あるいは他の受講生と比較が有効である．しかし先ほど述べたようにプログラミングへの門戸が広くなっているため，教師が受け持つ受講生数が増えつつあり，学習者が困難に直面した時点での支援を現状と同等にきめ細かく行うことが困難となりつつある．そのため，今後は教師の負担を減らしつつ，かつ生徒を適切にサポート・評価することが課題となってくる．

そこで，本稿では演習形式のプログラミングの授業において，コーディングの試行回数・更新頻度・実際の成績評価などのメタデータに着目し，これらを統計的に分析することで，コードを精読せずとも受講生のコーディング過程を評価できる可能性について議論する．メタデータをもとにコーディング過程を評価する意義として，生徒の理解度を知るために教師がコードを精読する手間が省けること，課題の完成形のみでなく，生徒の課題に対する取り組み方も新たに評価対象とできる可能性があることなどが挙げられる．

本稿の構成は以下の通りである. 2 章では関連研究について述べる．３章では実験をするにあたり，どのような授業で得られたどのようなデータをどう用いるのかについて述べる．4 章では扱ったデータ変数・分析結果を評価し，5 章ではまとめを述べる．

2 関連研究

プログラミング教育の支援をテーマとする研究はすでに多くなされており [2]，そのテーマは学習者支援，教師の授業進行支援，教師の採点支援の３つに分けられると考える．本章では教師の授業支援に関する研究・教師の採点支援に関する研究に着目し，関連研究と本稿との違いについて述べていく．

2.1 教師の授業進行支援に関する研究

円滑な授業進行のための受講生の理解度把握の支援は演習修了時に次の演習に向けて行うものと，演習中にリアルタイムでコーディング状況を分析するものの 2 種類ある．一方で，演習中にコーディング過程を分析し，支援を行う研究も多くなされている．藤原 [3] らは受講生のコンパイル回数，実行回数，エラー数，行数などの履歴を利用しリアルタイムで支援を行う手法を提案している．これは教師が事前にコーディング過程の各種履歴を定義することで，過度に遅れを見せるパターンに合致した受講生がいた場合，検出するというものである．これはあらかじめ教師が履歴のパターン化をしなければいけないという点で教師の負担となる．井垣ら [4] は解答プログラムの行数とプログラム作成開始から終了までの作業時間を元に受講生を相対的に比較することで，作業が遅れておりサポートが必要な受講生の早期検出を行う手法を提案している．

これらの関連研究はコードを精読することなく受講者の学習状況の把握をする方法を提案しており，コードを精読するので

はなくメタデータを統計分析することで受講生の過程評価の可能性を模索する本研究とは趣旨が異なる．

2.2 教師の採点支援に関する研究

教員が受講生 1 人 1 人のプログラムをチェックすることは大きな負担であり，その負担を軽減するための研究は数多くなされている [7] [8]．島袋 [5] らは受講生のプログラムを課題型プログラムと作品型プログラムに大別し，評価補助システムを提案している．課題型プログラムの条件を「(1) 入力を与えた場合求める値が出力される (2) 条件分岐される入出力にも対応 (3) 特定の命令を含んでいる」とし，この全ての条件に対応する採点補助システムを実装した．この研究では採点者があらかじめ入出力を記載したファイルを用意しておく必要がある．

小西ら [6] はプログラミング教育において教師を支援するプログラム評価機構を提案している．ここでは学習者プログラムの評価はプログラムの正誤だけでなく，教師が演習問題を出題する際の教育目標を満たすか否かに基づく必要があると述べられており，受講生が書いたコードが教師が望む教育目標を含んでいるかについて評価している．今回の実験で評価指標を検討し，有用性を実証することで，コンパイル数などに左右されず，幅広い言語を用いた演習に応用できる．

これらはコード量やコード構造などのメタデータを正解データと照らし合わせることで自動採点を行おうとしているが，本研究はメタデータを統計的に分析し，それらを必ずしも成績評価に直結させないという点で先行研究と異なっている．

3 分 析 対 象

本章では分析の対象となる演習の開講形態と，分析に用いるデータの種類と収集方法について説明する．

3.1 演習の内容と形態

本稿で検討対象とする授業は大学内で行われている「プログラミング基礎演習」という授業である．この授業は神戸大学国際人間科学部において開講されているものであり，97 人の受講生に対し教授 1 人で授業を行う．使用言語としては JavaScript のビジュアルコーディングライブラリである p5.js を用いた授業である．授業形式としては，教授が与えたプログラミング課題を受講生が実際に作成するという演習形式となっている．各プログラミング課題は図 1 のように最初に教授が作成した雛形が用意されており，これを元に受講生は課題作成を行う．つまり，課題作成にあたって，全ての受講生のスタートと目指すゴールが同じであり，他の受講生との到達度比較が可能であると言える．

受講生 97 人に対して教授は 1 人であるため，2 名の TA が行き詰っている受講生の元で実際にサポートしつつ教授がプログラミング課題を作るための技術解説を行うという流れで授業が進行される．課題評価についてはプログラムを記述するにあたってあらかじめ提示した条件を満たしているか・最終形が要求したものと相違ないかどうかで評価する．あらかじめ提示した条件をより多く満たせているものについては加点し，満たす

```
// 練習問題：神戸市のマーク
function setup(){
  createCanvas(400, 400);
  background(255);
  kobeCity(100, 200, 100);
  kobeUniv(300, 200, 100);
}

function kobeCity(x, y, size){
  push();
  noFill();
  strokeWeight(size * 0.25);
  strokeCap(SQUARE);
  arc(x + size * 0.25, y, size, size, QUARTER_PI * 3, QUARTER_PI * 3 + PI);
  // BLANK[1]
  pop();
}

// おまけ：神戸大学のロゴを作りたい人向けに色情報だけ置いておきます
function kobeUniv(x, y, size){
  let red = color(196, 0, 0);
  let blue = color(14, 47, 146);
  let green = color(22, 131, 46);
  let gray = color(77);
}
```

図 1　用意される例題

条件が少なかった場合はその分だけ減点するという方式で評価を行っている．これら各課題の評価に加え，予習課題の成績・平常点を加味した上で最終成績を算出する．この授業の受講者の特徴としてほとんど全員がプログラミング未経験者という点があげられる．これは今後小学校などで必修化されるプログラミング教育と環境が似通っており，今後のプログラミング教育必修化で予想される教員の負担増大の解消に本研究が寄与できる可能性がある．

3.2 収集データ

今回の分析ではプログラミング基礎演習を受講している全受講生 97 人を対象とする．学内ネットワーク上に学籍番号ごとに割り振られたフォルダが存在し，そのフォルダ内に課題ごとのフォルダが格納されているという階層構造になっている．そのフォルダ内の各プログラムを 40 秒ごとに取得し，プログラムが更新されていた場合は対象のファイルを取得するという形で今回のデータ収集を行った．このスパンでファイル取得を行なった理由は，生徒 97 人のプログラムが更新されているか全て確認するために 40 秒かかるためである．授業内ではテキストエディタである Atom を用いたプログラム作成を基本としていたが，P5.js の Web エディタを用いてプログラムを作成する生徒も見受けられた．このように Web エディタを用いた学生のコーディング試行回数は取得できないことになる．

4 分析の流れ

まず，分析の際に用いる変数として，各課題の点数及びその課題を作成した際のプログラム変更回数 ‥ 総合成績が挙げられる．また，これらを用いて以下の 2 つのアプローチにより分析を行う．

(1) 各課題の得点分布に着目した分析

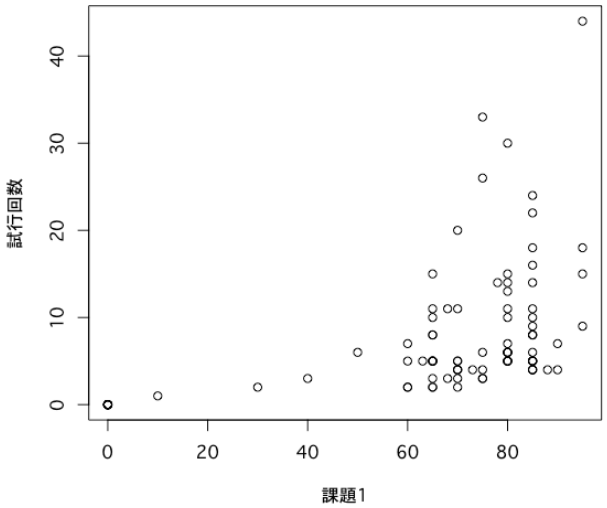

図 2　課題 1 における点数とプログラム変更回数の散布図
(相関係数:0.496, P 値:2.63e-07, 95%信頼区間:0.329-0.632)

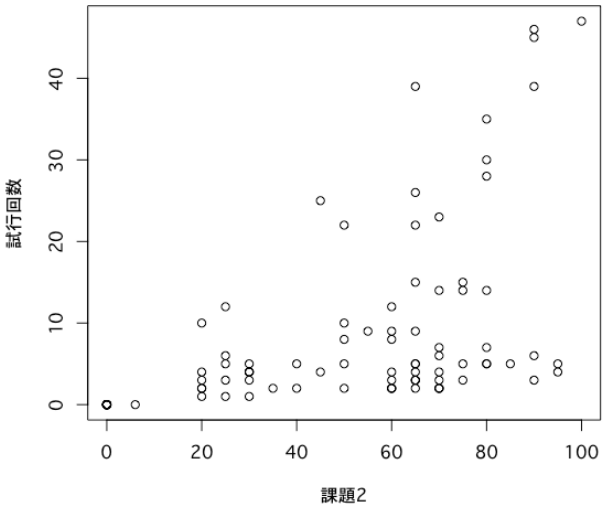

図 3　課題 2 における点数とプログラム変更回数の散布図
(相関係数:0.533, P 値:1.855e-08, 95%信頼区間:0.373-0.662)

(2) 受講生の各課題の得点推移に着目した分析

これら 2 つのアプローチによる分析をもとに，課題のメタデータからコーディング過程を評価できるかどうか検討する．

4.1　各課題の得点分布に着目した分析

各課題の得点分布に着目した分析では，課題ごとに着目し，ある課題における生徒らの点数とプログラム変更回数の相関を算出，無相関検定を実施する．これによりプログラム変更回数の多さは課題の点数に比例するのかを検証する．

4.2　受講生の各課題の得点推移に着目した分析

受講生の各課題の得点推移に着目した分析では，受講生ごとに着目し，受講生の各課題における点数とプログラム変更回数の推移を分析する．これによりプログラム変更回数が多さは難易度に比例するのかを検証する．

5　評　　価

5.1　各課題の得点分布に着目した分析についての評価

図 2，図 3，図 4 は課題 1〜3 について，課題点数を横軸，プログラム変更回数を縦軸に描画したグラフである．課題 1〜3 における課題点数ならびにプログラム変更回数を変数としてとり，これらの同時分布は 2 変量正規分布に従っていた．よって相関を調べた上で無相関検定を行った．

それぞれの課題についてプログラム変更回数と課題点数との相関を調べた結果，課題 1〜3 において課題 1，課題 2 における相関係数はそれぞれ 0.496，0.533 とどちらも正の相関が見られた．また，課題 3 については相関係数が 0.343 と弱い正の相関が見られた．これら 3 つについて無相関検定を行なった結果，3 つ全てで 95% 信頼区間において帰無仮説を棄却することができた．よって，弱くはない相関を認める．これにより，プログラム変更回数が多い生徒ほど得点が高い傾向にあると言える．

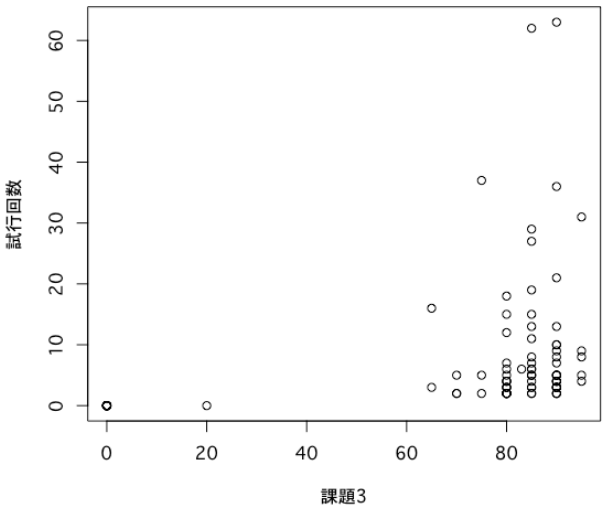

図 4　課題 3 における点数とプログラム変更回数の散布図
(相関係数:0.343, P 値:0.00056, 95%信頼区間:0.155-0.508)

5.2　受講生の各課題の得点推移に着目した分析についての評価

図 5 は受講生の得点およびプログラム変更回数の推移をグラフ化したものである．まず分析対象とする課題の難易度設定についてであるが，課題 1 群・課題 2 群・課題 3 群ともに最後の問題 (課題 1-3・課題 2-2・課題 3-2) の難易度が高く設定されている．よって，難易度が高い課題と低い課題のプログラム変更回数の変化を比較することで何らかの特徴的な動きを見出そうと試みた．結果として，グラフから読み取る限り，課題の難易度と変更回数の間に関係は特に見られないことがわかった．

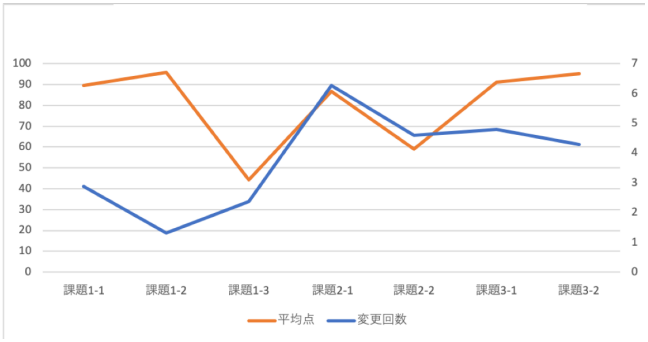

図 5　受講生の得点およびプログラム変更回数の推移

6　おわりに

今回プログラミング基礎演習における課題のメタデータを分析することでコードを精読せずとも演習課題の過程を評価できる可能性について検討した．今回の分析では主に 2 つの点が課題として上がった．1 つ目は保存されたファイルに対して 40 秒に 1 回のスクレイピングで変更を判断した点である。この条件下であれば，40 秒の間にコードを大量に変更しても，1 行のみ変更しても同じ 1 回としてカウントされてしまう．今後は，プログラム変更を自動で検知し，保存するようなエディタの拡張機能を実装し，生徒に使用してもらう必要がある．2 つ目は開発環境が様々であった点である．Atom 上で作業する生徒が大半であったが，Web エディタ上でコーディングし，コードをコピーペーストした生徒も見受けられた．今後の分析精度の向上のためにも，使用する開発環境の統一は課題となる．

各課題の得点分布に着目した分析では，それぞれの課題において生徒のプログラム変更回数と課題の点数に一定の相関があり，かつ有意差が認められた．一方で受講生の各課題の得点推移に着目した分析では，難易度が高い課題と低い課題のプログラム変更回数の変化を比較することで何らかの特徴的な動きを見出そうと試みたが，グラフから読み取る限り，課題の難易度と変更回数の間に関係は特に見られなかった．今回はプログラム変更回数と各課題の点数という 2 つのメタデータを元に各生徒の過程評価を行なった．今後はコード量の増減やプログラム更新スパンなどの他のメタデータを用いて，過程評価の可能性についてさらに検討していくことが重要である．

プログラミング基礎演習は演習という名を冠しているが，100 人規模を教師 1 名と TA でサポートするため、自習または講義のような形式になりがちなのが現状である．今回の分析結果を元にすると，プログラム変更回数が多い生徒はある程度自分で手を動かして課題に取り組むことができ，回数が少ない生徒は手がとまっており困っているという可能性を論ずることもできる．今回の結果を元に今後は，メタデータを活用した人手を多く必要としない大規模プログラミング演習の実践についても検討していきたい．

また、別のメタデータを用いて受講生の課題への取り組みを時系列で評価することで，生徒がそれぞれの課題に対して感じた難易度に応じたコンテンツ提供の可能性や，生徒が感じている難易度に応じてインストラクションを柔軟に対応する可能性についても模索していきたい．

さらに，プログラミング演習のメタデータを分析して得られた結果を元に，次回のプログラミング演習において過程の評価を試みることで，実際にこれらのメタデータが評価指標としてふさわしいのか，今回でた相関関係は年度ごとの受講生に依存するのかについて今後も評価していきたい．

7　謝　　辞

本研究の一部は科研費 (19K03000) の支援による．ここに記して謝意を表す．

文　　献

[1] 小学校プログラミング教育の手引 (第二版), 平成 30 年 11 月, 文部科学省.
[2] 加藤利康, 石川孝. ”プログラミング演習のための授業支援システムにおける学習状況把握機能の実現.” 情報処理学会論文誌 55.8 (2014): 1918-1930.
[3] 藤原理也,et al ”ストリームデータによる学習者のプログラミング状況把握.”DEWS2007 D9-5(2007)
[4] 井垣宏, et al. ”プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案.”情報処理学会論文誌 54.1 (2013): 330-339.
[5] 島袋舞子, et al. ”プログラミング課題・作品評価補助システムの提案” 研究報告コンピュータと教育 (CE) 2016.14 (2016): 1-9.
[6] 小西達裕, 鈴木浩之, 伊東幸宏, ”プログラミング教育における教師支援のためのプログラム評価機構.” 電子情報通信学会論文誌 D 83.6 (2000): 682-692.
[7] 田上恒大, 阿部公輝. ”比較的大きなプログラミング課題のための自動採点システム.” 情報処理学会研究報告コンピュータと 教育 (CE) 2006.16 (2006-CE-083) (2006): 135-140.
[8] 和田修平, and 井上潮. ”盗用発見と自動採点によるプログラミング演習課題の評価支援システム.” DEIM Forum. Vol. 2011. 2011