卒　業　研　究　報　告　書

令和7年度

研究題目

プログラミング講義中に現れる
補助が必要な学生の検出

指導教員　　上　野　秀　剛　　准教授

氏　　名　　明 松 龍 之 介

令和8年2月27日　提出

奈 良 工 業 高 等 専 門 学 校　　情 報 工 学 科

プログラミング講義中に現れる補助が必要な学生の検出

上野研究室　明松龍之介

教育現場におけるプログラミング教育では，座学と演習，課題を組み合わせた授業形式が一般的である．講義では，教員が新しい概念や構文を解説した後，受講者が実際にプログラムを作成することでスキルを習得する．しかし，少数の教員が多数の受講者を対象に講義を行う場合が多く，受講者一人ひとりの理解状況を把握し，適切な指導を行うことは困難である．この課題に対応するため，Online Judge System（OJS）などの自動採点システムが導入されているが，システムは採点のみを行うため，学習支援は十分ではない．近年，OJSに蓄積される提出ソースコードの履歴を分析し，「無作為修正」に着目してメトリクス分析を行う研究が報告されている．しかし，従来手法では最初の提出時刻を起点とした一定の時間幅で全ソースコードを対象にメトリクスを計算していたため，無作為修正と通常の修正が混在し，検出精度の低下と短時間での集中的な修正検出が困難という課題が存在した．本研究は，プログラミング講義中の学生に対してリアルタイムに補助の必要性を判断し，講義途中での学習支援を実現することを目的とする．先行研究で用いられた無作為修正検知のメトリクスを継承しながらも，計算方法を改善することで検出精度の向上を目指す．提案手法の主な特徴は，1)メトリクス計算の参照区間を直近の10分間に限定することで，無作為修正が行われた時間帯の特徴をより顕著に反映させることができる，2)参照区間を部分的に重複させることで，時間的な区分による連続提出の分断を回避し，短時間での集中的な修正行動を正確に検出できるという点である．本研究では，奈良工業高等専門学校情報工学科の本科3年生が受講したプログラミング講義における提出ソースコードを分析対象とした．2024年度と2025年度の2年間にわたり，合計60人の学生による計10,018ソースコードの提出履歴を収集した．実験では，従来手法と提案手法を同じデータセットに適用し，以下の3つのResearch Question（RQ）に基づいて比較検証を行った．RQ1）より短期間を対象に算出した無作為修正を検出するメトリクス群と未来のスコアの関係，RQ2）従来手法と提案手法で検出できる無作為修正の違い，RQ3）各メトリクスにしきい値を設けることの有用性，以上のRQ3つを問いとして設定する．RQ1の分析では，各メトリクスとスコアの相関を算出し，Welchのt検定を用いて統計的有意性を検証する方法で実施した．その結果，算出した4メトリクスのうち，3メトリクスで相関が見られた．RQ2の分析では，両手法で実際に無作為修正であると検出されたソースコードを目視で比較する方法で実施した．その結果，従来手法では長時間にわたる大量提出の様子を，提案手法では短時間で行われた部分的な修正の様子を無作為修正として検出した．RQ3の分析では，各メトリクスにしきい値を設定し，しきい値以上の学生としきい値未満の学生の平均スコアの差の有意性を確認する方法で実施した．その結果，全メトリクスで各学生群の平均スコアに有意差が見られ，補助が必要な学生を検出する上で，しきい値を設けることの有用性が示された．

目 次

1 はじめに

教育現場におけるプログラミング教育では，座学と演習，課題を組み合わせた授業形式が一般的である．座学や演習では，教員が講義を進行し，受講者はそれを聞きながら理解を深めたり，実際に手を動かしてプログラムを書くことでスキルを習得する．しかし，少数の教員が多数の受講者を対象に講義を行う場合，受講者一人ひとりの理解状況を把握すること，また把握した上で適切な指導を行うことは困難である．

課題の提出状況から受講者の様子を分析し，行き詰まる受講者の特徴を把握するための研究が複数存在する．先行研究では，各受講者のスナップショットから算出されるメトリクス値をもとに，誤答の原因を十分に理解せず修正と提出を繰り返す「無作為修正」を検知した[1]．この研究では無作為修正を行った受講者を検知できた一方で，講義時間内に提出された全てのソースコードをメトリクス値の計算に用いたことから，目的としていた演習時間内の無作為修正の検知には至っていない．

本研究は，長時間ソースコードを提出していなかったり，提出のたびにソースコードの大部分を変更するなど，一人では正答に辿り着けそうになく，補助を必要とする受講者のリアルタイムな検出を目的とする．ここでのリアルタイムとは，直近に提出されたスナップショットのみを分析に用いることで，時間経過で変化する補助を必要とする受講者を検出することを指す．本研究も同様に，スナップショットから算出されるメトリクス値を用い，受講者の状況識別を行う．仮説として，各受講者の課題の点数と先行研究で用いられた「課題を無作為に修正した度合いを示すメトリクス群」を時系列分析した結果に関係があると考えた．先行研究では，課題に取り組み始めてから10分ごとに，その時点までに提出された全ソースコードから指標を算出した．本研究では算出対象を直近の10分間に限定することで，より算出時点の状況を反映できる．また，算出を行う前後5分の計10分間も指標の算出対象に追加する．これにより，指標を算出するタイミングで時間的な区分が発生し，短時間での連続した課題の提出が検知できないという先行研究の課題を解決できる．

また，先行研究ではメトリクスにしきい値を設定しグループ分けすることで無作為修正の傾向を明らかにしていた．しかし，本研究ではメトリクス算出対象を直近の10分間のみに限定することでデータ数が減少することが予想される．データ数が減少することでしきい値の設定が困難になる可能性が考えられるため，本研究では各メトリクスにしきい値を複数設定し，しきい値設定の有用性を検証する．

本研究では，奈良高専情報工学科学生が受講した2024年度と2025年度分の講義における課題の提出履歴計60人分を分析対象として使用する．提案手法は従来

手法とは異なり，講義途中のデータから補助が必要な学生を特定可能であり，講義中の学生に対して理解状況に応じたリアルタイムな補助が可能である．本稿では以下の3つの問(Research Question)を立て，分析と考察を行い，提案手法が学生の理解状況の把握および教育支援に有用であるかを検証する．

- RQ1: より短期間を対象に算出した無作為修正を検出するメトリクス群と未来のスコアに関係はあるか?
- RQ2: 従来手法と提案手法で検出できる無作為修正に違いはあるか?
- RQ3: 各メトリクスにしきい値を設けることは補助が必要な学生の検出に有用か?

以下，2章では関連研究について説明し，3章で準備，4章で提案手法と実験設定を説明する．5章では実験の結果と考察を示し，6章では本研究のまとめと今後の発展について説明する．

2 関連研究

プログラミング教育支援に関する研究は，これまで数多く行われている．大野らはOnline Judge System(OJS) を用いたプログラミング教育において，誤答の原因を十分に理解せず修正と提出を繰り返す行動を「無作為修正」と定義した．OJSを使用した授業の提出履歴から修正行の偏りや短時間での提出回数などをメトリクス化することで無作為修正者を検知した[1]. 加藤はOJSのような教育支援システムに蓄積される学習履歴のデータが，学習状況の分析にどのように用いられているかを整理した[2]. その結果，受講者のシステム使用法の監視や分析，受講者ごとにコンテンツを変更する教育の適応化などを目的とし，文章を図表で表す視覚化などの方法によってデータ分析が実行されている例が多いことを明らかにした．加藤らはプログラミング演習において巡回だけでは把握が難しい学習状況を把握することを目的として，Webベースの授業支援システムに模範解答のプログラムを基準とした進捗判定機能を持つ学習状況把握機能を実装した[3]. 外れ値と判断されるしきい値を教員が適切に設定することにより，進捗に問題がある受講者の絞り込みが可能であることが示された．

プログラミング教育の支援を目的としたOJSを開発した研究も存在する．松永[4]，岩本ら[5]はプログラミング教育初期において，OJSを導入した事例を報告している．どちらの研究もスコアやアンケートなどからプログラミング教育に一定の有用性があるとしたうえで，受講者や課題ごとに効果の違い，実装した不正コピー検出の誤検知など今後の課題があることを示した．三野らはOJSを用いたプログラミング講義の受講者を対象に短時間の連続提出の有無と無作為修正を行ったかをアンケート形式で収集し，提出回数や行き詰まり率などのメトリクスの有用性を検証した[6]. その結果，メトリクスが無作為修正を検出した例がある一方で，誤検出という今後の課題があることが明らかになった．中川らは過去に提出されたソースコードをOJSに読み込ませることで，受講者の理解状況に適したソースコードを提示する学習支援の手法を提案している[7]. 被験者アンケートにより，有意に誤答の原因特定に役立つことが示された．清水らはOJSに提出されたソースコードを分析し，エラーの大部分を占める論理エラーを含むユーザごとのエラーの大規模な実態調査を行い，解決できた割合や種類の異なるエラー間の遷移，解決時間が長いエラーを特定した[8].

これらの既存研究は，OJSに関連した教育支援である点や，課題の進捗に不具合のある受講者の検出を目的とする点で本研究と類似する．しかし，本研究は1) 提出されたソースコードのみを対象に分析を行う点，2) 計算するメトリクスの時間的取り扱いが異なる点，3) 要補助者をリアルタイムで検出することを目的とする点で既存研究とは異なる．本研究により提出されたソースコードのみから補助が必要な受講者を検出できれば，講義中に受講者への指導を行うことが

可能となり, 既存研究とは異なるアプローチを介して受講者の単元ごとの理解度を向上できる.

3 準備

3.1 OJSを用いたプログラミング教育

本研究は高等教育で行われているプログラミング講義を対象とする. 一般的なプログラミング講義は1~2人の教員が数十人程度の受講者に対し講義を行う. 90分間の講義の前半では教員が資料を用いて解説を行ったり, プログラムを実際に動作したりして新たな単元の説明を行う. 講義の後半では, 演習として提示された課題に受講者が取り組み, 期限内に完成したソースコードを提出する. この授業形態では, 受講者が提出する大量のソースコードを少数の教員のみで採点しなければならない. また, 手動でソースコードを実行し, 実行結果を目視で確認する作業を繰り返さなければならず, 教員の負担が大きく, ヒューマンエラーによる採点ミスなどを誘発する恐れがある.

OJSを用いた講義の場合, 受講者が演習や課題で作成したソースコードをシステムに提出すると, 教員が用意したテストケースによってシステムが採点を行う. OJSを導入することで, 採点に伴う教員の負担を大幅に軽減し, ヒューマンエラーの発生を防ぐことができる. また, 受講者は採点結果を確認し, 100点でなければシステムからのフィードバックを確認してソースコードを修正し, 再び提出し100点を目指す. 採点がリアルタイムで受講者に通知されることで, 受講者は自らに間違いに気づき, 修正する機会が生じることで教育効果も高い.

3.2 無作為修正

無作為修正とは, プログラミング講義中に受講者が課題に取り組む際, エラーの原因を理解せずにコードを修正する行為を指す. 図1に, 無作為修正の例を示す. 図は課題として出題された問題と, 問題に対してある受講者が提出したソースコード群を示す. Rev.1~9は受講者が提出したソースコードの一部を示し, minは授業開始からの経過時間を表す. また, ソースコード内の薄い文字は直近の提出から変更されていないことを表す. この受講者は, リビジョン1の提出から10分間で計9回ソースコードを提出しており, Rev.9で正解している. リビジョン2からリビジョン9までの間で修正しているのはすべてprint文の中身であり, 短時間の間に正解とは異なる式を何度も当てはめている様子が分かる. これは, 問題の意図を理解していない, もしくは意図を考えることなく修正を繰り返していると考えられる.

このような受講者の行動は, OJSを用いたプログラミング講義でよくみられる. OJS上で出題された課題は採点結果をすぐに確認でき, また, 繰り返し提出が可能であるため, 作成したソースコードの正しさを自身で確かめずに"とりあえず"システムに提出する現象を引き起こす.

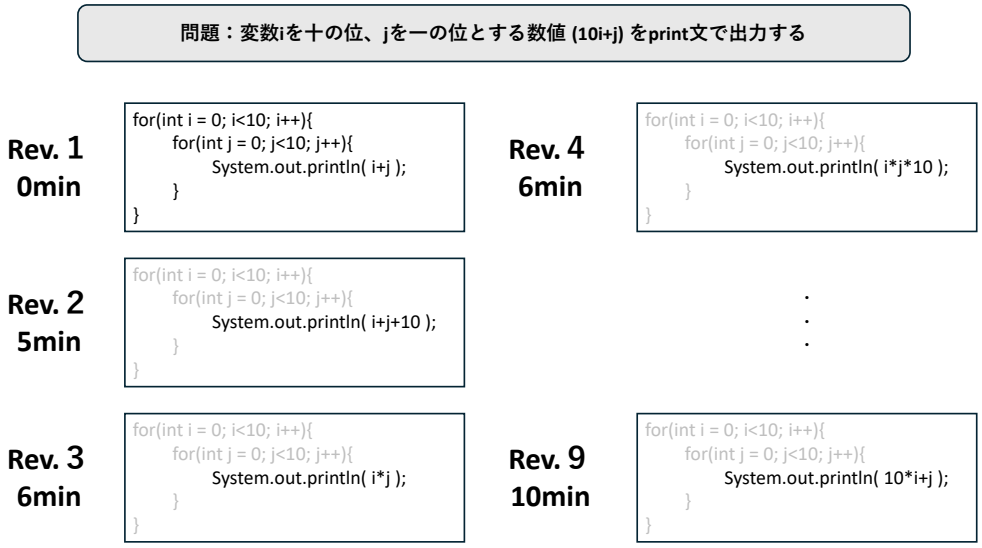

図 1 無作為修正の例

受講者が無作為修正を行うと，新単元の内容を理解することなく課題を終えてしまう恐れがある．無作為修正を行う受講者は，問題の意図を理解せずに修正を行うため，無作為修正によって100点を取った受講者は内容を十分に理解せずに課題を終える．プログラミングの講義では前回講義で学習した概念や構文が，次の講義では基本的な内容として扱われ，受講者が既に理解していることを前提に講義が進んでいくため，無作為修正者は次回以降のすべての講義の理解度が低下してしまう．

以上のような問題が発生するため，受講者の無作為修正を検知することはプログラミング教育効果を高めることに繋がる．先行研究では，最初の提出から分析時点までのすべてのソースコードを対象に分析を行った．しかし，無作為修正は演習に取り組んでいる間ずっと行われるわけではなく，エラー箇所が推定された段階で初めて行われ，また，修正が完了するか，繰り返しても点数が改善しないと判断すると停止すると考えられる．そのため，無作為修正をしていない時に提出されたソースコードを分析対象に含むことで，無作為修正の特徴量として不適切な評価を行っている可能性がある．

3.3 無作為修正のメトリクス

本研究では，先行研究で提案された無作為修正の特徴を表す4つのメトリクス [1] を用いる．各メトリクスは各課題に対する，受講者の初回の提出からt分間に提出されたソースコードの変更履歴から計算する．また，リビジョンkをR_kと表す．

- $Revs(t)$:t 分間のリビジョン数
- $Freq(t)$: 修正行の偏り
- $AveDiff(t)$:1 回あたりの平均修正行数
- $DiffLine(x,t)$:x 行以下の修正が続いた回数

3.3.1 $Revs(t)$

t 分間に提出されたリビジョン数を表す. 受講者が無作為修正を行う場合, 体系的な判断無しに自身の持つ修正パターンを適用するため, 修正によって改善する可能性が低く, 無作為修正をしない受講者と比較して一定時間内の修正・コンパイル回数が多くなる傾向があると考えられる. $Revs(t)$ は, OJS の提出記録から算出する.

3.3.2 $Freq(t)$

t 分間に修正された行の偏りを表す. 受講者が無作為修正を行う場合, エラーの原因を深く考えないため, エラーの原因と推測した特定の行に対して繰り返し修正を行う傾向があると考えられる. そこで, ソースコード各行に対する修正回数の分散を算出することで, t 分間に行われた修正が特定の行にどの程度集中しているか分析する.

$Freq(t)$ の算出は以下の手順で行う. まず, 初回提出から t 分間に提出されたすべてのリビジョン R_1, R_2,..., R_k,..., $R_{Revs(t)}$ 間の各行の対応を取り, ユニークな行に対して ID= $\{id_1,\ id_2,\ ...,\ id_i,\ ...,id_N\}$ を割り当てる. N はすべてのリビジョンを通じたユニークな行数を示す. 次に, 修正履歴から各行に対する修正回数 c_i と, その合計 S を求める.

$$S = \sum_{i=1}^{N} c_i \tag{3.3.1}$$

最後に, 各行に対する修正回数の分散 μ から $Freq(t)$ を算出する.

$$\mu = \frac{1}{N} \sum_{i=1}^{N} \frac{c_i}{S} \tag{3.3.2}$$

$$\mathrm{Freq}(t) = \frac{1}{N} \sum_{i=1}^{N} \left(\frac{c_i}{S} - \mu \right)^2 \tag{3.3.3}$$

3.3.3 $AveDiff(t)$

t分間に行われた修正の，1回あたりの平均修正行数を表す. 受講者が無作為修正を行う場合，エラーの原因と推測した特定の行のみに対して修正を行うため，1回あたりの修正行数が短くなると考えられる．そこで，修正履歴から求めた各リビジョン間の修正行数と$Revs(t)$から，受講者がどの程度の規模の修正を行っているかを分析する．

$$\text{AveDiff}(t) = \frac{\text{各リビジョン間の修正行数}}{Revs(t) - 1} \tag{3.3.4}$$

3.3.4 $DiffLine(x, t)$

t分間に行われた修正のうち，x行以下の修正が連続した回数の割合を表す. 受講者が無作為修正を行う場合，エラーの原因と推測した特定の行に対して繰り返し修正を行うため，少ない行に対する修正を繰り返すと考えられる．そこで，修正履歴からx行以下の修正が続いた回数の最大値を算出し，修正回数と合わせて，少ない行に対する修正がどんな頻度で行われているかを分析する．

$$\text{DiffLine}(x, t) = \frac{x\text{行以下の修正が続いた回数の最大値}}{Revs(t) - 1} \tag{3.3.5}$$

本研究では，$x = 1$として本メトリクスを用いる．

3.4 先行研究の課題と提案手法

先行研究の成果として，$Freq(t)$, $Revs(t)$, $DiffLine(1, t)$の3メトリクスにおいて，各10分区間のソースコード群における最新リビジョンの点数score(t)との相関が見られた．また，受講者をメトリクス値の大小やscore(t)の遷移の違いでグループ分けした分析により，$Freq(t)$と$Revs(t)$を組み合わせることで無作為な修正を繰り返す受講者を検出できることが分かった．

本研究では，3.2節でも述べた通り，無作為修正を行う受講者であっても常に無作為修正を繰り返しているわけではない点に着目する．先行研究では各課題に対する各受講生の最初の提出時間を0分とし，t分後$(t = 5, 10, 15, ..., 45)$の間に提出されたソースコードすべてを対象にメトリクスを算出している．図2に先行研究におけるメトリクス算出時の時間幅を示す. 図の白星はエラーに対する検討をした上での修正による提出を，黒星は無作為修正による提出を示す. 無作為修正が常に行われず，ある期間に集中して発生すると仮定した場合，先行研究の算出方法には2つの問題がある．

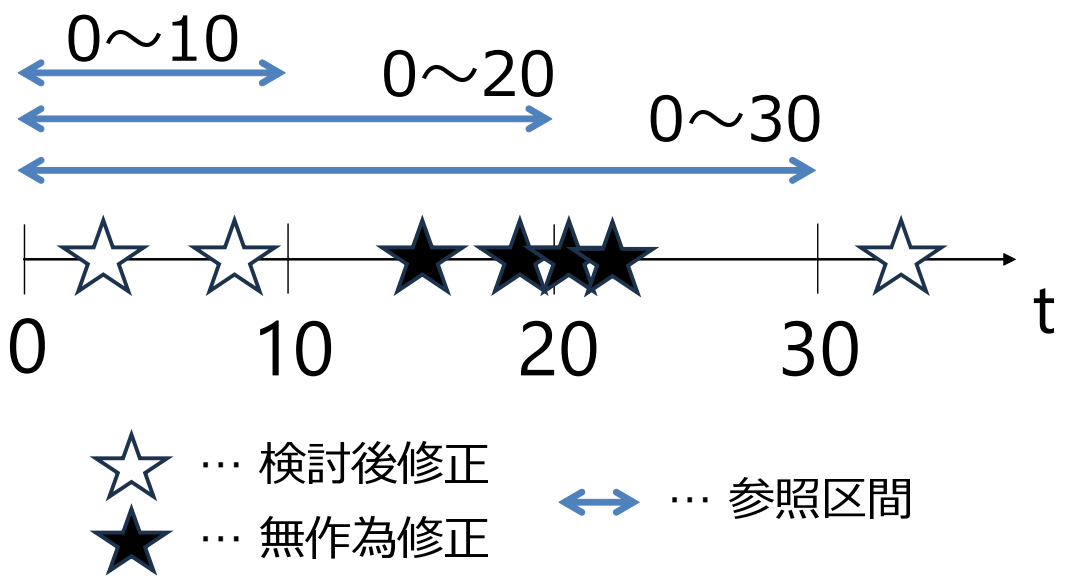

図 2 従来手法におけるメトリクス計算の時間幅

- 長い参照区間による希薄化
  先行研究の算出において，0~45 分などの幅が広い時間幅からメトリクスを算出した場合，区間内には無作為修正を行った時間と行っていない時間が混在する．そのため，t の値が大きくなる場合に無作為修正の様子がメトリクスの値に反映されにくくなる．無作為修正の様子がメトリクスの値に反映されにくいと，無作為修正がメトリクスの値にどのような影響を及ぼすかが曖昧で，メトリクスの有用性の検証ができない恐れがある．

- 参照区間による分割
  図 2 のように，最初の提出から 20 分段階付近で無作為修正が行われた場合，先行研究の算出方法では区間の区切りによって連続した提出が分断される．そのため，短期間に連続した提出がある事を検出するメトリクスが想定通りに計算されず，検出漏れを起こす恐れがある．

以上の議論を踏まえ，本研究ではメトリクスを計算する時間の区間を変更する方法を提案する．図 3 に提案手法のメトリクス算出方法の概要を示す．提案手法は 1) メトリクスを算出する区間の開始位置を移動して直近の 10 分間のみを参照する，また，2) 参照区間を部分的に重複させる．直近 10 分間のみを参照することで，無作為修正が行われた区間と行われていない区間でメトリクスの値に差が生じやすく，無作為修正の様子が従来手法よりも顕著に表れる．また，参照区間を重複させることで，区切り位置による連続提出の分断を回避する．

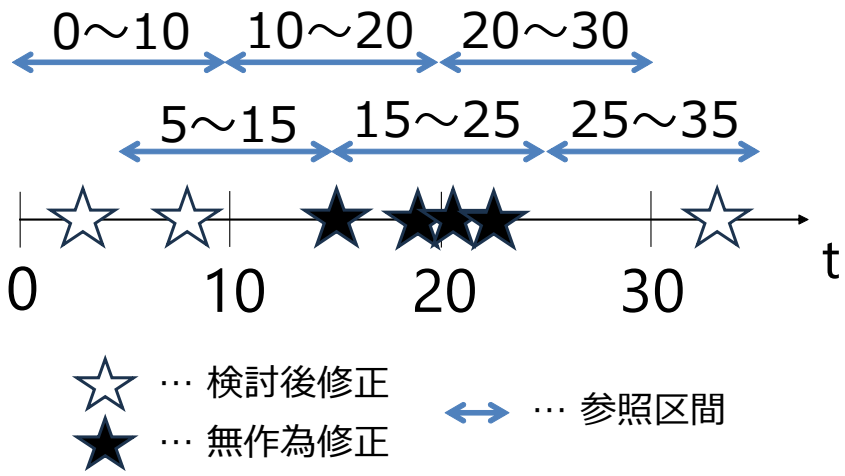

図 3 提案手法におけるメトリクス計算の時間幅

4 実験

提案手法の有効性を評価するために，奈良工業高等専門学校情報工学科の本科3年生が受講するプログラミングⅡで受講者が作成したソースコードを分析する．本講義はJava言語を対象としたプログラミング演習で，受講者が提出したソースコードが課題ID，受講者ID，リビジョン，提出日時，テストケース実行結果，scoreとともにOJSによって記録される．本研究について受講生に事前に説明し，データの利用に同意した受講生の提出のみを分析対象のデータセットとする．2024年4月17日～10月2日に収集された21人25課題計3767ソースコードと，2025年4月16日～11月4日に収集された39人42課題計7251ソースコードの合計10018ソースコードを対象とする．

先行研究の手法（従来手法）と本研究の提案手法の実験結果を比較するため，同じデータセットを用いて，3.4節で述べた従来手法と，3.5節で述べた提案手法それぞれの方法で3.3節のメトリクスを計算する．従来手法，提案手法共に$Revs(t) = 1$の場合はメトリクスの算出が不可能なため，分析から除外する．分析ではソースコード群を対象に3.3節で説明した各メトリクスを算出し，score(t)との相関を求めることで，RQ1”より短期間を対象に算出した無作為修正を検出するメトリクス群と未来のスコアに関係はあるか?”に回答する．また，分析対象の提出履歴に対して，提案手法でのみメトリクス値に差が見られた履歴や，従来手法では差が見られたにもかかわらず提案手法では差が見られなかった履歴について，その特性を考察することでRQ2:”従来手法と提案手法で検出できる無作為修正に違いはあるか?”に回答する．加えて，各メトリクスにしきい値を設定し，各しきい値によって分けられたソースコード群ごとにscore(t)の平均を算出する．しきい値以上のグループとしきい値未満のグループの平均score(t)の差を求め，Welchのt検定によってその差が有意であるかを調べることで，RQ3”各メトリクスにしきい値を設

けることは補助が必要な学生の検出に有用か?”に回答する．しきい値は各メトリクスごとに最大値の$10 \times n$ % ($n = 1 \sim 10$)の10個を用意し，補助が必要な学生な学生を検出するうえで特定の値をしきい値として設定する事が有用か考察する．

5 結果と考察

5.1 先行研究との比較

表1に従来手法と提案手法で計算した$Revs(t)$とscore(t)の相関を示す. 従来手法で計算した$Revs(t)$とscore(t)の相関係数は-0.105 ~ -0.226と弱い負の相関で, 全計測区間の半分程度のt区間で有意な相関が見られた($p < 0.05$). 提案手法で計算した相関係数は, -0.173 ~ -0.283と同様に弱い負の相関だったが, 有意な相関はあまり見られなかった. 以上の結果から, 3.3節での仮説通り, 一定時間内の修正・コンパイル回数が多い受講者のscore(t)が低いと解釈できる. また, 従来手法と同様の相関が提案手法でも見られたことから, $Revs(t)$のt区間を短時間に設定しても無作為修正の検出に繋がる可能性がある.

表2に, 従来手法と提案手法で計算した$Freq(t)$とscore(t)の相関を示す. 従来手法で計算した$Freq(t)$とscore(t)の相関係数は+0.156 ~ +0.229と弱い正の相関で, 対象人数が最も少ない0 ~ 5分以外のt区間で有意な相関が見られた($p < 0.05$). 提案手法で計算した$Freq(t)$とscore(t)の相関係数は+0.160 ~ +0.233と同様に弱い正の相関で, 全計測区間の半分程度のt区間で有意な相関が見られた($p < 0.05$). 以上の結果から, 3.3節での仮説とは異なり, 一定時間内の修正がわずかな行に集中している受講者のscore(t)が高くなると解釈できる. また, 従来手法と同様の相関が提案手法でも見られたことから, $Freq(t)$のt区間を短時間に設定しても無作為修正の検出に繋がる可能性がある.

表3に従来手法と提案手法で計算した$AveDiff(t)$とscore(t)の相関を示す. 従来手法で計算した$AveDiff(t)$とscore(t)の相関係数は-0.114 ~ -0.199と弱い負の相関で, 全計測区間の半分程度のt区間で有意な相関が見られた($p < 0.05$). 提案手法で計算した相関係数は-0.178 ~ -0.180と同様に弱い負の相関だったが, 有意な相関はほと

表1 $Revs(t)$における従来手法と提案手法の相関係数比較

従来手法				提案手法			
時間幅	分析対象人数	相関係数	p値	時間幅	分析対象人数	相関係数	p値
0~5	102	-0.103	0.305	0~10	145	-0.099	0.237
0~10	191	-0.226	0.002**	5~15	172	-0.124	0.106
0~15	289	-0.030	0.617	10~20	160	-0.173	0.029*
0~20	356	-0.107	0.043*	15~25	153	-0.283	0.000**
0~25	431	-0.086	0.074	20~30	152	-0.150	0.064
0~30	512	-0.105	0.017*	25~35	156	-0.038	0.639
0~35	575	-0.148	0.000**	30~40	160	-0.033	0.677
0~40	639	-0.160	0.000**	35~45	138	-0.246	0.004**
0~45	697	-0.197	0.000**	-	-	-	-

*: $p < 0.05$, **: $p < 0.01$

表 2 $Freq(t)$ における従来手法と提案手法の相関係数比較

従来手法				提案手法			
時間幅	分析対象人数	相関係数	p 値	時間幅	分析対象人数	相関係数	p 値
0∼5	102	0.111	0.269	0∼10	145	0.137	0.101
0∼10	191	0.169	0.019*	5∼15	172	0.233	0.002**
0∼15	289	0.159	0.007**	10∼20	160	0.216	0.006**
0∼20	356	0.212	0.000**	15∼25	153	0.158	0.050
0∼25	431	0.189	0.000**	20∼30	152	0.160	0.049*
0∼30	512	0.206	0.000**	25∼35	156	0.186	0.020*
0∼35	575	0.229	0.000**	30∼40	160	0.118	0.139
0∼40	639	0.177	0.000**	35∼45	138	0.157	0.066
0∼45	697	0.156	0.000**	-	-	-	-

*: $p < 0.05$, **: $p < 0.01$

表 3 $AveDiff(t)$ における従来手法と提案手法の相関係数比較

従来手法				提案手法			
時間幅	分析対象人数	相関係数	p 値	時間幅	分析対象人数	相関係数	p 値
0∼5	102	-0.041	0.681	0∼10	145	-0.180	0.030*
0∼10	191	-0.090	0.218	5∼15	172	-0.044	0.564
0∼15	289	-0.034	0.565	10∼20	160	-0.077	0.335
0∼20	356	-0.098	0.065	15∼25	153	-0.092	0.259
0∼25	431	-0.170	0.000**	20∼30	152	-0.105	0.196
0∼30	512	-0.171	0.000**	25∼35	156	-0.132	0.100
0∼35	575	-0.168	0.000**	30∼40	160	-0.152	0.055
0∼40	639	-0.114	0.004**	35∼45	138	-0.178	0.037*
0∼45	697	-0.199	0.000**	-	-	-	-

*: $p < 0.05$, **: $p < 0.01$

んど見られなかった．以上の結果から，3.3 節での仮説通り，一定時間内の平均修正行数が多い受講者の score(t) が低くなり，また参照区間を縮小した場合にも同様の解釈が可能であることが分かる．

両手法ともに $AveDiff(t)$ と score(t) との相関が弱く，また有意差も少ないため，メトリクスとしての有用性が限定的である，もしくは相関を求める分析が有効でないことが示唆される．

表 4 に従来手法と提案手法で計算した $DiffLine(1,t)$ と score(t) の相関を示す．従来手法で計算した $DiffLine(1,t)$ と score(t) の相関係数は+0.122 ∼ +0.201 と弱い正の相関で，対象人数が最も少ない 0 ∼ 5 分以外の t 区間で有意な相関が見られた (p < 0.05)．提案手法で計算した相関係数は+0.181 ∼ +0.268 と少し弱い正の相関で，ほぼすべての区間で有意な相関が見られた．以上の結果から，3.3 節での仮説とは異

表 4 $DiffLine(1,t)$ における従来手法と提案手法の相関係数比較

従来手法				提案手法			
時間幅	分析対象人数	相関係数	p 値	時間幅	分析対象人数	相関係数	p 値
0∼5	102	0.038	0.707	0∼10	145	0.089	0.285
0∼10	191	0.143	0.049*	5∼15	172	0.181	0.017*
0∼15	289	0.122	0.039*	10∼20	160	0.203	0.010*
0∼20	356	0.134	0.011*	15∼25	153	0.196	0.015*
0∼25	431	0.156	0.001**	20∼30	152	0.202	0.013*
0∼30	512	0.200	0.000**	25∼35	156	0.268	0.000**
0∼35	575	0.201	0.000**	30∼40	160	0.098	0.216
0∼40	639	0.147	0.000**	35∼45	138	0.191	0.025*
0∼45	697	0.140	0.000**	-	-	-	-

*: $p < 0.05$, **: $p < 0.01$

なり, 一定時間内における1行以下の修正が多い受講者のscore(t)が高くなると解釈できる. また, 従来手法と同様の相関が提案手法でも見られたことから, 計測区間を縮小した場合にも同様の解釈が可能であることが示される. 特に提案手法の25 ∼ 35分区間では相関係数が+0.268と最大となり, 1行以下の修正の割合が高いほどscore(t)が高いという関係が顕著に表れている.

表1 ∼ 4より, 提案手法で計算したメトリクスとscore(t)の相関は従来手法の相関と大きな違いがないことが分かる. これは, 区間を10分間としたことで, 従来手法では困難だった短時間の提出を無作為修正である認識できた一方で, 10分以上時間が離れた提出を無作為修正だと認識できなくなったことが原因と考えられる. しかし, 従来手法と比較して提案手法で計算したメトリクスと各score(t)との相関係数の絶対値が大きくなっており, 補助が必要な受講者の検出に有効である.

また, 従来手法と比べて提案手法は有意な相関が少ないことが分かる. これは, 10分間ごとにメトリクス計算を行う関係で, 従来手法と比べて提案手法が対象に取ることのできるデータ数が少なくなってしまったことが原因と考えられる.

図4に, 従来手法と提案手法の両方で高いメトリクスとなったソースコードを示す. この提出履歴は, 従来手法と提案手法ともに $Revs(t) = 4$, $Freq(t) = 0.540$, $AveDiff(t) = 1$, $DiffLine(1,t) = 1$ となった. 図におけるリビジョン3では, 11行目の変数に代入する値を変更している. しかし, この修正はプログラム的に意味のない修正であり, エラーの原因を深く考えていない修正, すなわち無作為修正を行っていると判断できる.

従来手法と提案手法それぞれで検出できる提出履歴の差を比較する. 図5に提案手法でのみ無作為修正であると判定されたソースコードを示す. この提出履歴は, 従来手法では $Revs(t) = 3$, $Freq(t) = 0.489$, $AveDiff(t) = 1$, $DiffLine(1,t) = 1$ となり, 提案手法では $Revs(t) = 2$, $Freq(t) = 1$, $AveDiff(t) = 1$, $DiffLine(1,t) = 1$ となった.

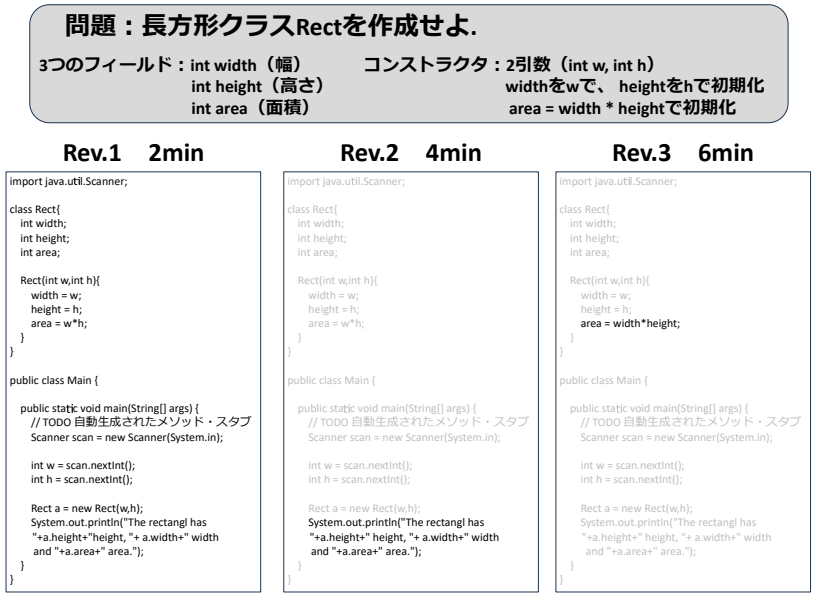

図4両手法で高メトリクス値となったソースコード

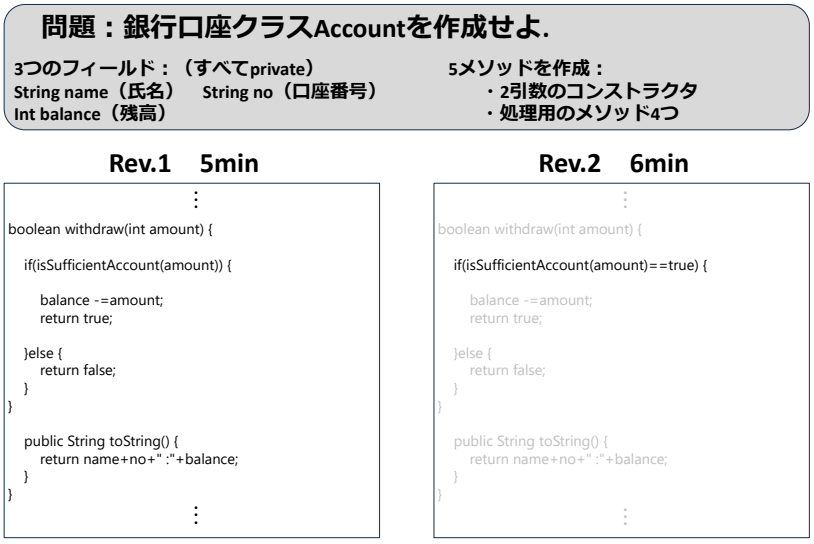

図5提案手法でのみ無作為修正であると判定されたソースコード

修正内容を見ると，初回の提出から1分後にif文の条件式だけを変更して再提出している．しかし，この修正はプログラムの動作に何も影響しない修正であり，エラーの原因を深く考えずに修正していると判断できる．このような修正履歴に対して，従来手法では参照区間が長いことで$Freq(t)$の値が小さくなったことにより，無作為修正として検出することができなかった．一方，提案手法は直近10分間のみの参照によって$Revs(t)$の値が大きくなることを防いだため，$Freq(t)$が高い値となり無作為修正として検出することができた．

図 6 に従来手法でのみ無作為修正であると判定されたソースコードを示す. この提出履歴は, 従来手法では $Revs(t) = 7$, $Freq(t) = 0.261$, $AveDiff(t) = 2.667$, $DiffLine(1,t) = 0.167$ となり, 提案手法では $Revs(t) = 3$, $Freq(t) = 1$, $AveDiff(t) = 4.5$, $DiffLine(1,t) = 0$ となった. 修正内容を見ると, 3 分の間で変数 tf に代入する値を何度も変更している. しかし, これらの修正はエラーに関係するものではなく, エラーの原因を正しく理解せずに修正していると判断できる. この受講者は, 図 6 で挙げたソースコードを含め, 講義時間内にソースコードを 7 つ提出しており, 従来手法ではこの $Revs(t) = 7$ という値から無作為修正として検出した. しかし, 提案手法では 10 分区間でメトリクスを計測しており, $Revs(t) = 3$ と大きな値が出なかったことで無作為修正として検出しなかった. $Revs(t)$ は従来手法と提案手法を併用することで, 無作為修正として検出する範囲が拡大できる可能性がある.

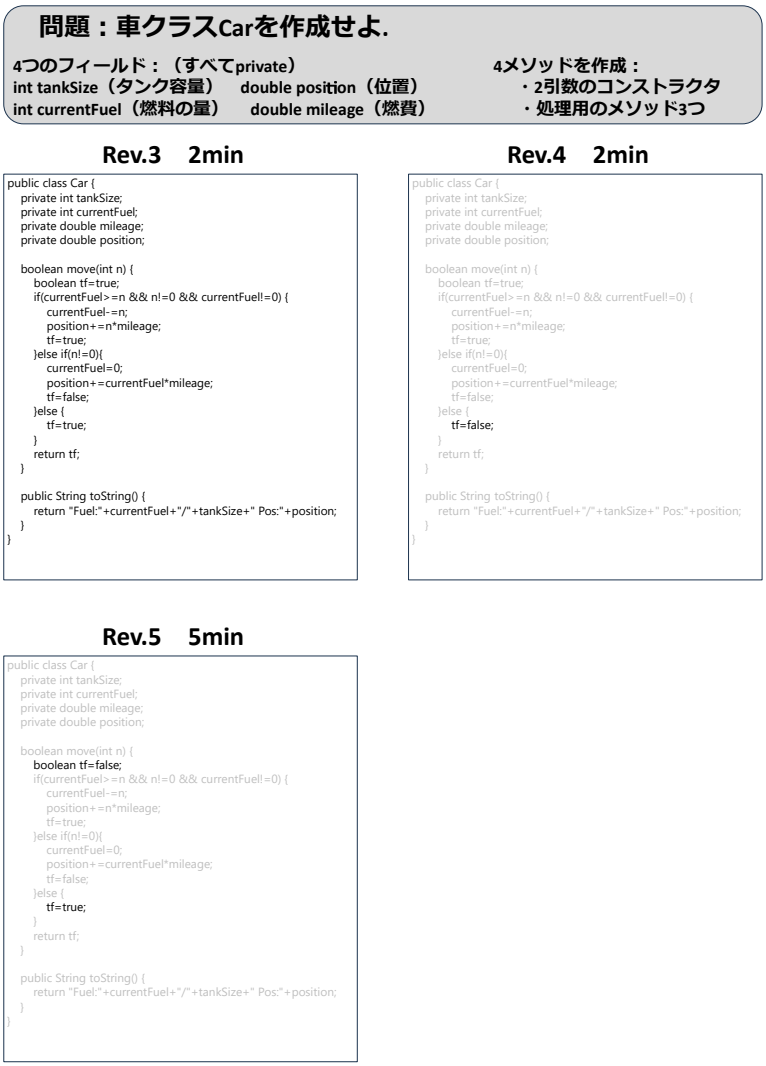

図 6 従来手法でのみ無作為修正であると判定されたソースコード

以上の結果から，RQ1, RQ2に回答する．

RQ1: より短期間を対象に算出した無作為修正を検出するメトリクス群と未来のスコアに関係はあるか？

RQ1への回答：$Revs(t)$, $Freq(t)$, $DiffLine(1,t)$の3メトリクスでscoreと相関が見られた．

RQ2: 従来手法と提案手法で検出できる無作為修正に違いはあるか？

RQ2への回答: 従来手法では，長い参照区間により長時間にわたるソースコードの大量提出の様子を，$Revs(t)$によって無作為修正として検出することができた．しかし，提案手法では参照区間を10分間に限定したために，10分を上回るような長時間にわたる提出の様子を無作為修正として検出することができなかった．一方，従来手法では無作為修正として検出できなかった短時間の部分的な修正の様子を，提案手法では参照区間の短さから$Freq(t)$によって無作為修正として検出することができた．

5.2 しきい値に基づいたscore(t)の比較

表5に$Revs(t)$のしきい値S_rと，しきい値を上回った受講者と下回った受講者の平均スコアを示す．S_rが18以上の時，しきい値を超えるソースコードの件数が極端に少ないため，以降の分析の対象から除外する．S_rが15以下のすべての場合でS_r以上のscoreがS_r未満より低く，9.0〜22.9点の差があり，S_rが3, 9, 15の場合で有意差が見られた．各S_rで平均スコアに差が見られた理由として，エラーについて理解している受講者は提出回数が少なく，エラーについて理解していない受講者は提出回数が多かったことが考えられる．分析の結果から，S_rを設定して$Revs(t)$の値を観測することは，補助が必要な受講者の検出に有用であると考えられる．

表5 $Revs(t)$のしきい値と各受講者群の平均スコア

S_r	S_r以上		S_r未満		score差	p値
	件数	平均score	件数	平均score		
3	742	62.7	494	73.4	-10.7	0.000**
6	261	63.2	975	68.0	-4.8	0.064
9	111	58.8	1125	67.8	-9.0	0.014*
12	43	55.6	1193	67.4	-11.8	0.062
15	29	44.6	1207	67.5	-22.9	0.003**
18	7	54.3	1229	67.0	-12.7	0.407
21	4	50.0	1232	67.0	-17.0	0.473
24	4	50.0	1232	67.0	-17.0	0.473
27	2	80.0	1234	67.0	-23.0	0.632
30	2	80.0	1234	67.0	-23.0	0.632

*: $p < 0.05$, **: $p < 0.01$

表6に$Freq(t)$のしきい値S_fと，しきい値を上回った受講者と下回った受講者の平均score(t)を示す．S_fがすべての場合でS_f以上のscoreがS_f未満より高く，11.5〜15.3点の差があり，S_fがすべての場合で有意差が見られた．各S_fで平均スコアに差が見られた理由として，エラーについて理解している受講者はエラー箇所のみを修正し，エラーについて理解していない受講者はエラー箇所以外の複数個所の修正が多かったことが考えられる．分析の結果から，S_fを設定して$Freq(t)$の値を観測することは，補助が必要な受講者の検出に有用であると考えられる．

表7に$AveDiff(t)$のしきい値S_aと，しきい値を上回った受講者と下回った受講者の平均スコアを示す．S_aが77.1以上の時，しきい値を超えるソースコードの件数が極端に少ないため，以降の分析の対象から除外する．S_aが67.5以下のすべての場合でS_a以上のscoreがS_a未満より低く，7.6〜19.4点の差があり，S_aが67.5以下のすべての場合で有意差が見られた．各S_aで平均スコアに差が見られた理由として，問題について理解している受講者のエラーは些細なミスによるものが多く，かつエラー箇所のみ修正するため修正行数が少なくなると考えられる．一方，エラーについて理解していない受講者はエラー箇所を探すため，様々な箇所を修正するために修正行数が多かったことが考えられる．分析の結果から，S_aを設定して$AveDiff(t)$の値を観測することは，補助が必要な受講者の検出に有用であると考えられる．

また，全ソースコードの5%程度を占める$S_a = 57.84$以上のソースコードについて，一度の修正で57行以上の修正をする例は以下が考えられる．

1. 100行以上のソースコードを作成する必要がある課題に取り組んでいる場合
2. 提出するソースコードを取り違え，全く異なるソースコードを提出した場合

1.から考慮すべき点として，$AveDiff(t)$で実際に補助が必要な受講者を検出す

表6 $Freq(t)$のしきい値と各受講者群の平均スコア

S_f	S_f以上		S_f未満		score差	p値
	件数	平均score	件数	平均score		
0.1	576	73.1	660	61.6	11.5	0.000**
0.2	465	76.1	771	61.5	14.6	0.000**
0.3	396	77.4	840	62.1	15.3	0.000**
0.4	341	77.5	895	63.0	14.5	0.000**
0.5	235	79.3	1001	64.1	15.2	0.000**
0.6	227	79.1	1009	64.3	14.7	0.000**
0.7	226	79.0	1010	64.3	14.7	0.000**
0.8	226	79.0	1010	64.3	14.7	0.000**
0.9	226	79.0	1010	64.3	14.7	0.000**
1.0	226	79.0	1010	64.3	14.7	0.000**

*: $p < 0.05$,　**: $p < 0.01$

表 7 $AveDiff(t)$ のしきい値と各受講者群の平均スコア

S_a	S_a 以上		S_a 未満		score 差	p 値
	件数	平均 score	件数	平均 score		
9.64	669	63.0	567	71.8	8.8	0.000**
19.3	510	62.5	726	70.1	7.6	0.001**
29.0	345	59.8	891	69.8	10.0	0.000**
38.6	248	55.2	988	70.0	14.8	0.000**
48.2	160	52.0	1076	69.2	17.2	0.000**
57.8	73	51.4	1163	68.0	16.6	0.000**
67.5	26	48.0	1210	67.4	19.4	0.007**
77.1	12	46.7	1224	67.2	20.5	0.059
86.8	2	100	1234	66.9	33.1	0.000**
96.4	1	100	1235	67.0	33.0	-

*: $p < 0.05$, **: $p < 0.01$

る上で，課題で作成するソースコードの規模ごとにS_aを変える必要がある．2.から考慮すべき点として，提出するソースコードを取り違えた受講者を補助が必要な受講者であると誤検出してしまうため，S_aが極端に大きい場合は除外するなどの工夫が必要である．

また，全ソースコードの半分以上が$AveDiff(t)$が10未満であったため，S_aが大きくなるほど分析対象が減少してしまった．総じて，S_aの決定方法についても今後の発展だと考える．

また，$AveDiff(t)$とscore(t)には線形的な強い相関は見られなかったものの，S_aが大きくなるほどS_a以上とS_a未満の両グループでscore(t)が低下する傾向が観測された．本結果は，$AveDiff(t)$はscore(t)と線形的な強い相関はなく，score(t)の大小を直接予測する指標ではないが，あるしきい値以上でscore(t)の分布が変化する，非線形な関係を示していることを示唆する．

表8に$DiffLine(1,t)$のしきい値S_dと，しきい値を上回った受講者と下回った受講者の平均スコアを示す．S_dが0.7以上の時，同じソースコードを分析していることから，以降の分析の対象から除外する．S_dがすべての場合でS_d以上のscoreがS_d未満より高く，13.3〜21.8点の差があり，S_dがすべての場合で有意差が見られた．各S_dで平均スコアに差が見られた理由として，想定通り問題を理解している受講者は些細なミスによるエラーが多く，エラー箇所の1行のみ修正し，エラーについて理解していない受講者はエラー箇所を探すために修正行数が多かったことが考えられる．分析の結果から，S_dを設定して$DiffLine(1,t)$の値を観測することは，補助が必要な受講者の検出に有用であると考えられる．

また，S_dが大きくなるほどS_d以上とS_d未満のグループ間でscore(t)の差が大きくなる傾向が観測された．本結果は，各学生のすべての修正に対する1行以下の修正の割合が大きい学生ほどscore(t)が高くなることを示唆しており，他のメトリ

表 8 $DiffLine(1,t)$ のしきい値と各受講者群の平均スコア

S_d	S_d 以上		S_d 未満		score 差	p 値
	件数	平均 score	件数	平均 score		
0.1	289	77.2	947	63.9	13.3	0.000**
0.2	274	78.0	962	63.9	14.1	0.000**
0.3	243	79.1	993	64.0	15.1	0.000**
0.4	216	80.0	1020	64.3	15.7	0.000**
0.5	214	80.2	1022	64.2	16.0	0.000**
0.6	152	85.3	1084	64.4	20.9	0.000**
0.7	147	86.2	1089	64.4	21.8	0.000**
0.8	147	86.2	1089	64.4	21.8	0.000**
0.9	147	86.2	1089	64.4	21.8	0.000**
1.0	147	86.2	1089	64.4	21.8	0.000**

*: $p < 0.05$, **: $p < 0.01$

クスよりscore(t)との相関が強いという5.1節の結果と一致する.

以上の結果から, RQ3に回答する.

RQ3: 各メトリクスにしきい値を設けることは補助が必要な学生の検出に有用か?

RQ3への回答: $Revs(t)$, $Freq(t)$, $DiffLine(1,t)$, $AveDiff(t)$ のすべてのメトリクスで, しきい値を設けることでスコアの低い受講者を検出できた.

6 おわりに

本研究は，プログラミング講義中の学生に対してリアルタイムに補助が必要な学生を検出することを目的に，提出されたソースコードからメトリクスを計算し分析を行った．

1つめの実験では，従来手法で計算したメトリクスとスコアの相関を，提案手法で計算したメトリクスとスコアの相関と比較した．実験の結果，すべてのメトリクスで従来手法と同様の相関が見られた．また，全メトリクスでスコアとの相関係数の絶対値が大きくなり，補助が必要な学生の検出に有用であることが示唆された．

2つめの実験では，各メトリクスごとにしきい値を設定し，しきい値を上回った受講者と下回った受講者の間に，スコアの平均点で有意な差が表れるかを確認した．実験の結果，全メトリクスで各受講者群のスコアの平均点に有意な差がみられたため，しきい値を設けることは補助が必要な学生の検出に有効であることが示唆された．

本研究の結果から，各メトリクスを計算し，しきい値を設けることで補助が必要な学生を把握し，指導を行うことで教育効果を高められると考えられる．

本研究の今後の課題として，参照区間の変更が挙げられる．従来手法には，先行研究の結果と比較して相関係数の傾向が変わったため，データセットの性質に左右されるという疑念が残った．提案手法には，参照区間を10分間としたことで，10分以内に2つ以上のソースコードが提出されていないとメトリクスを算出できないという欠点がある．また，従来手法と比較して分析対象のソースコード数が少ないため，従来手法から有意な相関の数が減少してしまった．以上から，2つの手法を組み合わせることで各手法の欠点の改善が期待できる．

留意すべき点として，各手法で要補助として検出されたソースコードを目視で確認を行ったが，要補助として検出されたソースコードすべてを確認したわけではない．そのため，検出されたソースコードのどの程度が真に補助を必要としている受講者であるかを分析することで，メトリクスの有意性がより明らかにできると考えられる．

また，本研究はメトリクスにしきい値を設けることが補助が必要な学生を検出に有用であるかを確認する目的でしきい値を設けた．しかし，しきい値の設定方法には特に根拠がないため，理論に基づいてしきい値を設定し分析を行うことは本研究の改善のいえる．

本研究は，講義時間内に補助が必要な学生を検出する目的で行ったが，提供していただいたソースコードを目視で確認したところ，支援を必要とする学生はそもそも講義時間内にソースコードを提出していない傾向があった．また同様に，作成するソースコードの規模が大きい，難しい課題ほど時間外に取り組まれて

いる傾向にある．以上のことから，講義時間外に提出されたソースコードに対象に分析を行い，補助が必要な学生を検出することは本研究の発展であるといえる．

謝辞

本研究を進めるにあたり, 終始にわたって多大なるご指導とご助言を賜りました上野秀剛准教授に, 心より感謝申し上げます.

また, 本研究に対して貴重なご助言をいただきました松村寿枝教授に, 厚く御礼申し上げます.

さらに, 本研究におけるデータ取得にご協力いただいた学生の皆様(計60名)に, 深く感謝いたします.

最後に, 本論文の査読において丁寧かつ的確なご指摘を賜りました岡村真吾教授に, 心より御礼申し上げます.

参考文献

[1] 大野優，上野秀剛，内田眞司:”ソースコードのスナップショットに基づいた無作為修正者の検出”，信学技報教育工学研究会，Vol. 118，No. 214，pp. 53–58 (2018).

[2] 加藤利康:”授業支援システムにおける学習分析の展開”，情報処理学会研究報告，Vol. 2014-CE-124，No. 23 (2014).

[3] 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，Vol. 55，No. 8，pp. 1918–1930 (2014).

[4] 松永賢次:”導入プログラミング教育におけるオンラインジャッジシステムの活用の試み”，情報科学研究，No. 31 (2010).

[5] 岩本舞，中村真人，小島俊輔，中島卓雄:”不正コピー検出手法を備えたオンラインジャッジシステムの開発”，情報処理学会論文誌　教育とコンピュータ，Vol. 1，No. 4，pp. 38–47 (2015).

[6] 三野天羽，上野秀剛:”自動採点システムを用いた講義における無作為修正者の提出行動分析”，信学技報教育工学研究会，vol. 121，No. 232，pp. 31–36 (2021).

[7] 中川尊雄，藤原新，畑秀明，松本健一:”プログラミング学習者向けソースコード提示システムTAMBA”，ソフトウェアエンジニアリングシンポジウム (2016).

[8] 清水翔太，槇原絵里奈，吉田則裕:”オンラインジャッジシステムにおけるエラーの組み合わせと解決時間時間の実態調査”，ソフトウェアエンジニアリングシンポジウム (2024).