Vol.2021-CE-162 No.14
Vol.2021-CLE-35 No.14
2021/12/4

プログラミングログ分析による支援の必要な学生の検知指標の提案とフィードバックツールの開発

井川 一渓[1,a)]　谷口 雄太[3,b)]　峰松 翼[2,c)]　大久保 文哉[2,d)]　島田 敬士[2,e)]

概要：
プログラマの需要が拡大する現代では，プログラミング教育が広く実施されている．学習者がコーディングを行うにあたって発生する多くの困難は，彼らの学習意欲の低下や脱落を引き起こす恐れがある．それを防ぐためには，教員による適切な支援が必要であるが，教員が人力で学生のコーディング状況を把握するには限界がある．そこで本論文では，コーディングログを分析して，コーディングにおいて支援すべき学生を検知するための指標を提案する．この目的は，教員が学習者の状況を把握することを容易にして，いち早く適切な支援を行えるようにすることである．この指標を検証した結果，改善の余地はあるものの，支援の必要な学生を検知できる可能性を示した．またそれと同時に，学生の過去のコーディングログを確認できるウェブアプリケーションの開発報告も行う．

1. はじめに

近年，社会全体において急速な IT 化と，それに伴ったプログラマ需要の増加が加速している．その影響により，日本では 2020 年に全国の小学校においてプログラミング教育が必修化されるなど，教育現場でも IT 化を意識した変革が見られている．このようなプログラミング教育需要の拡大の結果として予想されるのが，指導教員側の負担の増大である．

プログラミング学習を成功裏に終わらせるためには，学習意欲の高さが重要とされている [1], [2]．しかし，プログラムを書く上では多くの困難が伴なうため [3]，適切な支援なしでは学習意欲の低下，ひいては脱落に繋がる恐れがある．

そのため，教員は常に学習者のコーディング状況を把握し，支援すべき学生を発見することが重要になってくる．しかし，学習者一人ひとりの状況を人力で把握することは，学習者の人数が増えるに従って困難なものになるだろう．

このような学習者の状態把握を支援する方法の代表例はダッシュボードを通じた情報提供である [4], [5]．ダッシュボードは学生個人やクラス全体についての情報を視覚的に提供することで，短時間での状況把握を可能にするため，教員の負担軽減に繋がる．しかしこれらのダッシュボードでは，それぞれの学生がどのようにプログラムを書いたり，発生したエラーを解決しようとしているかを端的に把握することまではできていない．

そこで本研究では，以上の様な困難を容易なものとすることを主な目的とする．具体的には，プログラミングを行っている学生のログを分析することにより，コーディングにおいて問題が発生している学生を検知するための指標を提案する．ここでいう問題とは，例えば，エラーが頻発している，または, エラーを解決するためのコードを記述するのに時間がかかっているなどの状況を指す．

また，それとは別に，学生のコーディング状況をひと目で確認することができる，教員向けのフィードバックツールの開発を行う．これら２つの機能が，人力では困難な学生のコーディングのパフォーマンスの把握を容易にすることで，プログラミング講義における教員の助けになるであろう．

2. 支援すべき学生の検知

本節では，プログラミングを行っている学生のログを分析することで，コーディングに問題を抱え支援を必要としている者を検知するための指標の提案と検証を行う．

[1] 九州大学 工学部　電気情報工学科
[2] 九州大学 大学院システム情報科学研究院
[3] 九州大学 情報基盤研究開発センター
[a)] igawa@limu.ait.kyushu-u.ac.jp
[b)] taniguchi.yuta.941@ait.kyushu-u.ac.jp
[c)] minematsu@limu.ait.kyushu-u.ac.jp
[d)] fokubo@ait.kyushu-u.ac.jp
[e)] atsushi@ait.kyushu-u.ac.jp

Vol.2021-CE-162 No.14
Vol.2021-CLE-35 No.14
2021/12/4

図 1 WEVL のユーザインタフェース

2.1 事前設定

本研究においてプログラミングのログを収集する際に，WEVL というオンラインプログラミングツールを使用する．このツールは，我々が開発しているウェブアプリケーションであり，インターフェースとしては図 1 の通りである．WEVL は九州大学の一部のプログラミング講義において実際に採用されているものであり，これを使用してプログラミングを行うことで，ユーザが記述したコードやその記述時間，実行結果などの詳細な情報が自動的に専用のデータベースに保存されるというものである．

この WEVL を使用してユーザがコードを記述している際に，タイプが 3 秒停止した時点でオートセーブとして自動的にコードや時刻がデータベースに保存される．ただしこのイベントは，前回保存されたコードとの変化があった場合のみ発生するものである．また，ユーザがコードを実行した際にも同様の情報がデータベースに格納されるが，その場合は実行結果も同時に保存される．

本論文ではこの WEVL データベースから収集したデータを使用して実験を行う．また，この実験データは，2020 年の九州大学におけるあるプログラミング講義の内，3 回の講義の演習時間内のログを収集したものである．

2.2 指標の決定

2.1 節で述べた通り，WEVL データベースにコードが保存される場合，その記述した者の氏名や，コード，記述した日時などの情報が保存される．また，それが実行時の保存であった場合，実行結果も同時に保存される．ここで，コーディングにおいて問題が発生している学生を検知したいとき，これらのうちどの情報を指標として用いれば良いかを考察する．

一つ目の候補として考えられるものが，コードの長さである．例えば，長期的にコーディングが進んでいないといった状況は，コードの長さがほとんど変化していないといった特徴を発見すれば，即座に検知できると考えられる．また，コードを実行してもなかなか上手く行かず，何度もコードを書いて消すという操作を繰り返しているといった状況も，コード長増減を調べることで検知できると考える．

これを検証するために，学生の記述したコードの長さの変化をウェブ上で確認できるツールを作成した．実際にそのツールを動作させた画面が図 2 である．

画面上部の時間区分を指定して APPLY のボタンを押下することで，その時間内にコードが保存された学生のコードサイズが，学生ごとに線グラフで描写される．ここで，折れ線グラフの点一つ一つがセーブポイントを表しているが，このセーブポイントについて，自動保存であったときは灰色，実行保存されてかつその時に成功していれば緑色，その時に失敗していれば赤色になっている．また，グラフが一部破線になっている箇所は，セーブポイントの時間間隔が 3 分以上離れている場合である．そして，このセーブポイントを表す点を押下したときは，その時点で保存されたコードがポップアップで表示されるようになっている．

このツールを使用して，実際に講義内でコーディングを行った学生のコードとコード長の変化の観察を行った．

上で述べた通り，コーディング中に発生している問題の例として，長期的にコーディングが進んでいないということが挙げられる．このような状況下にある学生は，コードの長さがなかなか変化しないという特徴を見つけることで発見できると考えていた．しかし，実際には，演習として出された課題をすぐに書き終わってしまい，コード長が変化していない学生も検知してしまうという問題が見受けられた．

また，コードサイズの増減によって，コードを実行しても成功することができず，コードを書いて消すという操作を繰り返している学生を判別できると考察していた．しか

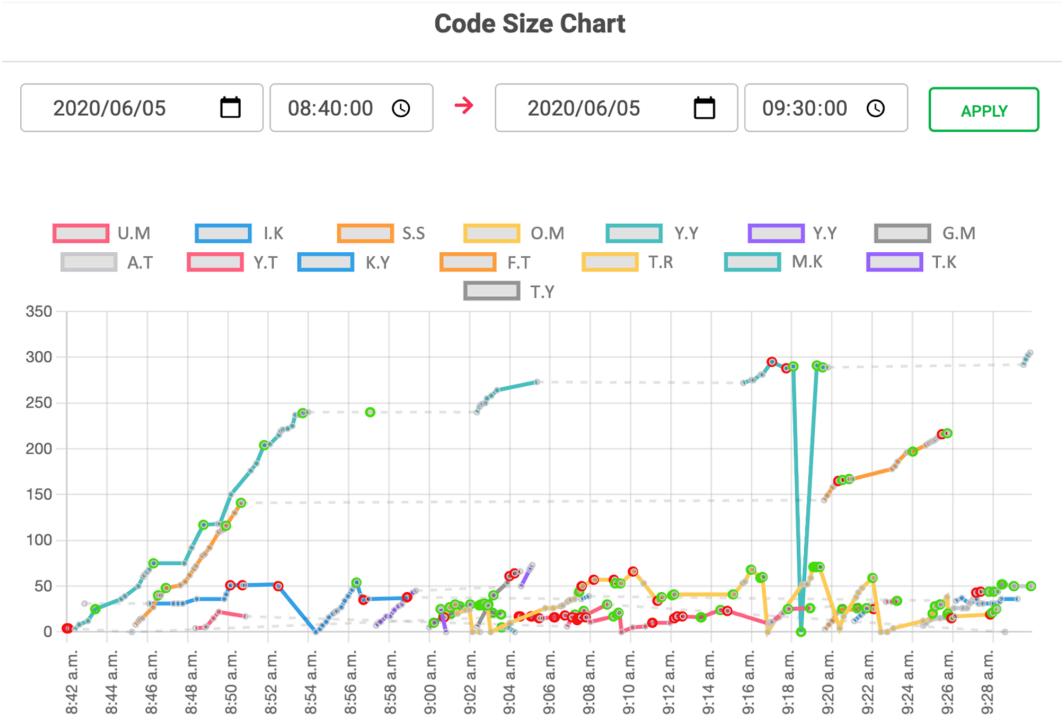

図 2 学生別のコードサイズの変化

し，ここでも問題点が見受けられた．それは，学生ごとのコードの書き方の違いである．例えば，講義内で複数の演習問題を課されたときに，その課題のコードを立て続けに書く学生や，課題が完成するたびに，また白紙から書き始めるといった学生が混在していた．こういった状況下では，コードが間違っていたから消去したのか，課題を完了したから消したのかを区別することは困難であろう．

以上の問題点から，今回の実験ではコード長を指標として採用することを取りやめた．

ここで，指標の候補として次に挙げられるものが，セーブポイントの種類である．上記の通り，WEVL データベースに保存されているセーブポイントは自動保存，実行してかつ成功，実行してかつ失敗の 3 種類が存在する．これを指標として使用すれば，失敗が立て続けに起こっている学生などを直接的に検知できるのではないかと考えた．よって，以降はこれを指標として設定し，検証を行っていく．

2.3 セーブポイントによるコーディング状況の推定

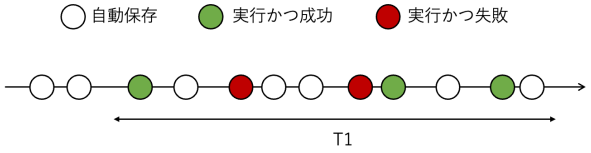

図 3 時系列順のセーブポイント

図 3 はある学生のセーブポイントを時系列に並べた場合を考えたものである．それぞれ，白色の点は自動保存を，緑色の点は実行保存かつ成功した場合を，赤色の点は実行保存かつ失敗した場合を表している．ここで時間区分 T1 を設けたときに，その区間内でのそれぞれのセーブポイントの割合を出すことができる．例えば図 3 の場合では，T1 内に，自動保存が 5 個，実行かつ成功が 3 個，実行かつ失敗が 2 個であるので，自動保存率が 50%，成功率が 30%，失敗率が 20%となる．ここで T1 が長くなればなるほど，その区間内のセーブポイントも多くなり，十分な量のデータを取る事ができるとわかる．しかし同時に，T1 が長すぎると，上記の割合が古いデータの影響を受け，本当に今現在困窮している学生を見逃してしまう恐れがある．よって本実験では，それらの兼ね合いを考慮して，時間区分 T1 を 30 分として設定している　次に，自動保存率と失敗率の割合が，それぞれ多い学生と少ない学生とでの状況の違いを考察していく．しかしその前準備として，自動保存率の特徴を確認しておきたい．

図 4 は実験データから算出した，自動保存率とセーブポイントの数の関係を描写したグラフである．このグラフから，自動保存率が高くなるにつれて，セーブポイント数も多くなるという全体的な傾向がある程度見られる．ここでセーブポイントの多さとは時間区分 T1 内でのコーディング従事度と捉えることもできる．つまり自動保存率が高いほど，長期的にコーディングを行っている可能性が高いと考えられる．

それを踏まえた上で，自動保存率と失敗率の割合の大小で考えられる学生のコーディング状況を推定していく．表

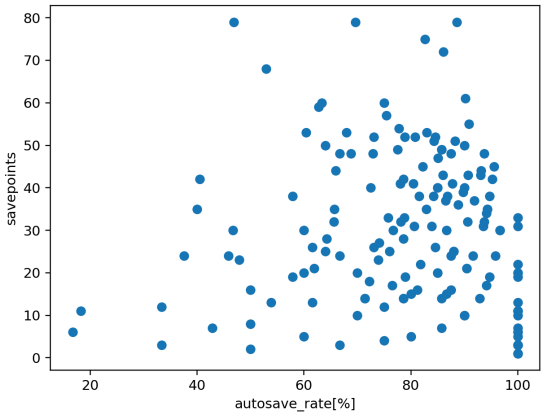

図 4 自動保存率とセーブポイント数の関係

表 1 セーブポイント割合によるコーディング状況の分類

		自動保存率	
		高	低
失敗率	高	グループ A	グループ B
	低	グループ C	グループ D

1 は実際にそれらの割合の大小から，コーディング状況を 4 種類に分類したものである．

まずグループ A を見ていく．ここの学生群は自動保存率が高い，つまり実行を頻繁に行うことなくコーディングを行っている集団であるといえる．その中でも比較的失敗率が高い学生は，現在失敗に対処するコードを記述中である可能性が高い．先述の通り，自動保存率の高さは長期的にコーディングを行っている可能性を示す．つまりこの集団は，長期的にエラーを含むコードを繰り返し記述している，比較的問題を抱えている可能性が高い，つまり危険度の高い学生群であると予想できる．

次にグループ B を見ていく．これは自動保存率が低い，つまり頻繁に実行を行っている集団であり，その中でも比較的失敗率が高い学生群である．この集団内の学生は，細い点を編集して何度も実行しているが失敗が続いているという状況が想定できる．

またグループ C,D であるが，これらはどちらも失敗率が低い集団で比較的安全であると予想できる学生群であるが，両者には自動保存率に違いが見られる．グループ C は比較的自動保存率の高い学生が属しているので，このグループは長期的にエラーの無いコードを記述している，比較的危険度が低い集団であると予想できる．

また，グループ D は，時間区分 T1 内ではあまりコーディングを行っておらず，失敗率も自動保存率も低いので，必然的に成功率だけが高い集団となる．よってこの集団の学生のコーディング状況として，数回コードを実行させて，何度も成功しているといった状況が考えられる．

2.4 閾値の設定

ここで実際に，自動保存率，失敗率の高低を区分するための閾値を設定する．この閾値を決定するにあたって週間テストの得点を参照する．週間テストとは，今回データを収集した講義で 1 週間毎に行われたテストであり，講義で受講した内容が問題として出題され，5 点満点で採点されるものである．まず，2.1 節で述べたデータに対して，自動保存率が高いグループ A と C，自動保存率が低いグループ B と D の学生群でそれぞれ，週間テストの平均点の差をとり，その点数差がなるべく大きくなるような閾値を考える．

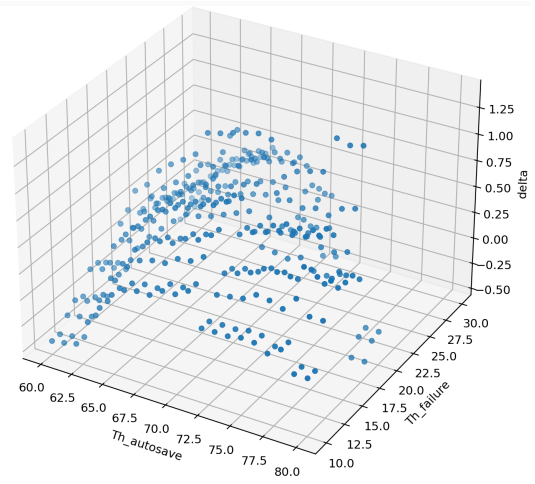

図 5 2 つの閾値とグループ間の点数差

図 5 は自動保存率と失敗率，それぞれの閾値を変えたときに，グループ A と C，グループ B と D の平均点の差がどのくらいであったかを表したグラフである．ここで，横軸の Th_autosave は自動保存率の閾値，Th_failure は失敗率の閾値，delta はグループ A と C，グループ B と D の平均点の差の合計である．本実験では，この delta を最大とするような自動保存率と失敗率の閾値とすることにした．その結果，自動保存率，失敗率の高低を分ける閾値はそれぞれ 70%，20%となった．

2.5 学生群の考察

閾値を定めたところで，次は実際に 4 つのグループに分類した結果を用いて．それぞれの群に属する学生の傾向を考察する．まずは，2.1 節で使用したデータを自動保存率の閾値が 70%，失敗率の閾値が 20%で分類し，その結果別れたグループごとに，週間テストの点数の分布表を作成した．今回はテストのスコアを用いてグループごとの傾向を考察しているが，本研究の目的は成績の悪い学生を予測するのではなく，コーディングに問題を抱えている学生を検知することであり，必ずしも危険度の高い学生が，テストの得点も低いとは限らないことに注意されたい．

設定した閾値で分類されたグループごとの分布表が図 6

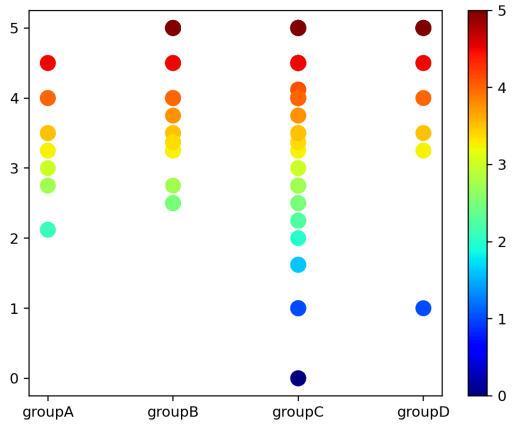

図 6 各分類の得点分布

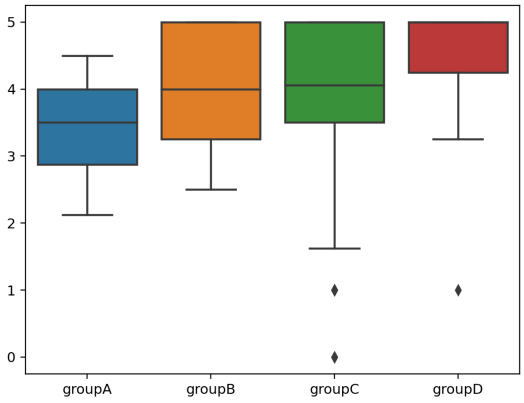

図 7 各分類の得点の箱ひげ図

と図 7 である．それぞれのグラフの縦軸は，週間テストでの最大 5 点の得点を表している．そして, その得点分布についての平均と標準偏差をまとめた表が表 2 である．

表 2 各得点分布の統計データ

	A	B	C	D	全データ
データ数	11	25	114	15	165
平均	3.38	3.88	3.92	4.38	3.92
標準偏差	0.83	0.92	1.13	1.10	1.09

この表から見て取れることの一つは，グループ A の得点の平均が，全体平均より比較的低い値を取っていることである．しかし，グループ A と同じように失敗率が高いグループ B では，全体平均と若干の差はあるが，グループ A ほどの有意な差は見られなかった．これについて考察を行う．まず，グループ A と B の違いは自動保存率であるが，これは 2.3 節で述べた通り，コーディングの従事度の差とも言いかえる事ができる．つまり，グループ B の学生はグループ A の学生ほど区間 T1 内においてコーディングを行っていない傾向があるといえる．このことから，グループ A は長期的にコーディングを行ったときにエラーを起こしやすい学生群であるといえるので，その傾向に伴ってテストの得点も他のグループより比較的に低くなっているのではないかと考えられる．また，グループ B の学生については，十分なセーブポイントがないので，失敗率が高いからといって，常に悪いパフォーマンスをしているとはいえない．よって危険度が成績に結びついていないのではないかと考察する．

また，グループ C,D は失敗率が低く，危険度は低いと思われる学生群であった．グループ D については全データに対して比較的高いテスト成績を取っている事がわかる．グループ D については，2.3 節の通り，そもそも時間区分内で十分な時間コーディングを行っておらず，数回実行させて何度も成功しているという状況を想定していた．そしてこのグループの得点が他に比べて比較的高いことから，このグループ内の学生は，短い時間で正しいコードを記述する事ができる集団であったのではないかと考察する．

それに対して，グループ C であるが，2.3 節では，長期的にコーディングを行っていてかつ，失敗率が低いので，比較的危険度の低い集団であると予想していた．しかしテストの得点を見てみると，全体データとほとんど差異がないことが見て取れる．この理由を考察してみる．ここで，2.1 節での定義で，セーブポイントの数から，自動保存率，成功率，失敗率に分けたことを思い出されたい．つまり 3 つの割合を足せば必ず 1 となるということである．それを踏まえると，グループ C は自動保存率ですでに多くの割合を専有しているので，失敗数が比較的多くなったとしても失敗率の値としては大きくはならないことがわかる．またこのような傾向が見られる状況として，コーディングを行っている学生がほとんど実行ボタンを押さずにコーディングを行っていることも考えられる．いずれにしても表 2 から分かる通り，グループ C のデータ数は 114 である．これは全体のデータ数の 7 割近くを占める値であり，グループ C についてはさらなる分析が必要であるといえる．

2.6 検知指標についての総括

今回，危険度の高い学生を検知するたの指標を考案し検証を行ったが，2.5 節で述べた通り，さらなる改良が必要である．この指標を実用段階まで持っていくには，グループの分類手法を見直した上で，別日や別講義のデータにも同様の分析を重ねる必要となるであろう．また，2.5 節の序文で述べた通り，必ずしも危険度とテスト成績が一致するとは限らないという前提がある．結果的に成績が良い学生であっても，コーディングにおいて一時的であっても困窮する場面は存在するはずであり，今後はそういった状況まで判別できるような指標に改善していきたい．

3. コーディングログモニタリングツール

本節では，学生が過去に記述したコードを時間遷移形式で表示することができるツールを紹介する．このツールには，コマ送り動画を表すタイムラプスとタイピングをかけ

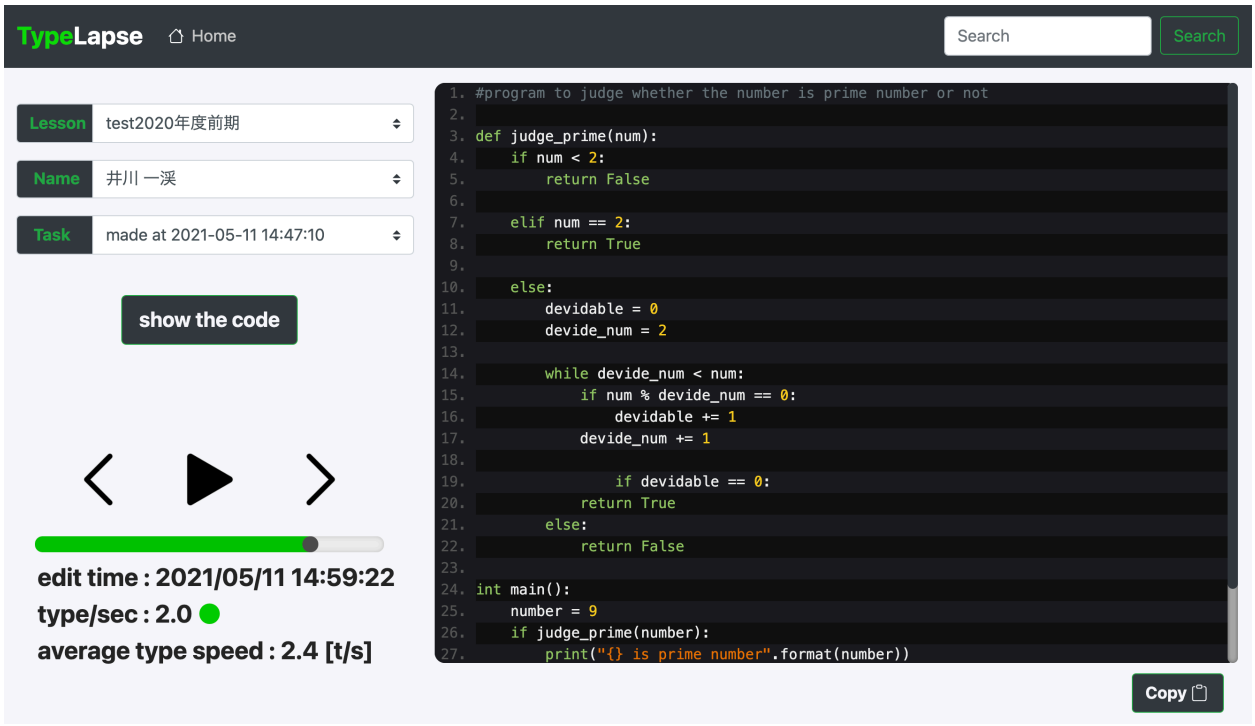

図 8 TypeLapse のユーザインタフェース

合わせ TypeLapse と命名している．

3.1 ツールの設計背景

プログラミング教育を行っている教員は，学生のパフォーマンスを確認する手段の一つとして，学生が記述中のコードを実際に見るという方法がある．そうすることで，初歩的な文法ミスをしてしまっている者や，なかなかコードを書けずにいる者を認知することができる．しかし，同時に複数人を教えるとなると，リアルタイムで学生全員のコードを視認することは困難であり，困っている学生を見落としてしまうこともあるだろう．

こうした問題の解決策の一つとして，学生のコードを過去に遡って確認するという方法が考えられる．WEVL を使ってコーディングした場合は，コーディング中に自動保存がなされ，そのコードと時刻が WEVL データベースに格納される．よって，教員がある学生のコードを過去に遡って確認したいときはこのデータベースを参照することになる．しかし，そのためには WEVL データベースの構造を理解していて，かつ，適当な SQL 文を記述する必要がある．この操作をウェブアプリケーションとしてより容易なものにするのが本ツール，TypeLapse である．

3.2 基本操作

TypeLapse のユーザインタフェースは図 8 の通りである．画面左側のセレクタで講義名，学生名，その学生が作成したセッションの順で一つのプログラミングセッションを絞り込む事ができる．それらを選択後，セレクタの下部にあるボタンを押すと，画面右側のエリアに選んだセッションの最新のコードが表示される．このとき，画面左下側に再生ボタンとレンジバーが表示される．ユーザはこのレンジバーを動かす，または，戻る，再生，進むボタンを押下することで，選択したコードを過去のセーブポイントに遡って見ることが可能になっている．

3.3 コピーアンドペーストの検知

ここで図 8 のレンジバー下部にある type/sec 2.0 について説明する．ここで type/sec とは，現在選択中のセーブポイントと 1 つ前のセーブポイントを比較し，1 秒間に何文字タイプを行ったかを表した値である．この数値を出すために，1 つ前の状態から何秒で何文字タイプして選択中の状態になったのかを知る必要がある．ここで，タイプ数であるが，これは前後のコードの編集距離を使っている．また，WEVL でコーディングを行う場合，キーボードをタイプしている間は，データベースに 1 秒毎にタイプ中というイベントが保存される．つまり，前後のセーブポイントの間に保存されたタイプ中のイベントの個数がそのまま，編集のためタイピングを行った秒数を表している．このように算出された 2 つの値を使用することで，type/sec という値を計算して表示している．

また，図 8 と図 9 に見られる通り，type/sec の横にはカラーマーカーが表示されている．これは type/sec の値によって緑，黄，赤と変化していくものであり，type/sec が 0 以上 6 未満であれば緑，6 以上 14 未満であれば黄,14 以上であれば赤色を示すようになっている．つまり，これ

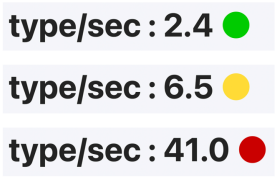

図 9 type/sec とカラーマーカー

は type/sec，1 秒間のタイプ数が多すぎることを表すマーカーであり，色が黄，赤と変わるにつれて，コピーアンドペーストをした可能性が高いことを表している．よって教員はこのマーカを確認することで，その学生がコピーアンドペーストを行ったかどうかを即座に知ることができる．

4. まとめと今後の方針

本論文では，プログラミング教育を行う教員が，学生のパフォーマンスを把握することを目的として，コーディングにおいて問題が発生している学生を検知するための指標の提案と，過去のコーディングログを確認できるウェブアプリケーションの開発を行った．検知指標については，閾値を設定することで 4 つの学生群に分け検証を行った．しかし，その結果は 2.6 節で述べた通り，グループ A，B，D については期待通りの結果が得られたが，グループ C についてはうまく分類できていると考えづらい得点分布となってしまった．そのため，閾値の設定手法から見直す必要もあると考える．また閾値とは別のパラメータとして，時間区分 T1 もあった．本論文ではこれを 30 分に固定して検証を進めてきたが，この値も様々な長さに変更しながら実験を重ねる必要があるだろう．また時間区分に関して，今回は T1 という一つだけの区分であったが，これを複数個にしてセーブポイントの時間変化を見ていくという手法も検討している．また，今回はコードの長さを指標として採用することを取りやめたが，今回採用したセーブ点の種類と組み合わせて分析することで，より高い精度で，危険度の高い学生を検知できるのではないかと考えている．以上のように，この指標は設定段階での手法の改善と，その後のさらなる検証が必要である．こうした検証実験を重ねた上で，実際の講義で実験運用できるものにしたいと考えている．

また，それと同時に，危険度が高いと検知した学生を教員に知らせるツールも必要である．現在はリアルタイムで生徒のコーディング状況を監視することができ，危険度が高いと検知した学生もそこにアラートされるような，ダッシュボード形式のウェブアプリケーションも開発中である．またこのアプリでは，指定した学生の過去の行動ログも確認できるようにする予定であり，今回開発した TypeLapse の機能もそこに盛り込まれることになる．さらなる研究を重ね，このウェブアプリケーションが完成すれば，プログラミングの講義を行う教員の大きな助けになるのではないかと期待している．

謝辞 本研究は，JST AIP 加速課題 JPMJCR19U1，科研費基盤研究（A）JP18H04125，および科研費若手研究 JP21K17863 の支援を受けた．

参考文献

[1] Jenkins, T.: The motivation of students of programming, *Proceedings of the 6th annual conference on Innovation and technology in computer science education*, pp. 53–56 (2001).
[2] Law, K. M., Lee, V. C. and Yu, Y.-T.: Learning motivation in e-learning facilitated computer programming courses, *Computers & Education*, Vol. 55, No. 1, pp. 218–228 (2010).
[3] Robins, A., Rountree, J. and Rountree, N.: Learning and teaching programming: A review and discussion, *Computer science education*, Vol. 13, No. 2, pp. 137–172 (2003).
[4] Fu, X., Shimada, A., Ogata, H., Taniguchi, Y. and Suehiro, D.: Real-time Learning Analytics for C Programming Language Courses, *Proceedings of the Seventh International Conference on Learning Analytics & Knowledge*, LAK '17, New York, NY, USA, ACM, pp. 280–288 (online), DOI: 10.1145/3027385.3027407 (2017).
[5] Diana, N., Eagle, M., Stamper, J., Grover, S., Bienkowski, M. and Basu, S.: An Instructor Dashboard for Real-Time Analytics in Interactive Programming Assignments, *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*, LAK '17, New York, NY, USA, Association for Computing Machinery, p. 272^^e2^^80^^93279 (online), DOI: 10.1145/3027385.3027441 (2017).