コーディング履歴を用いたプログラミング学習状況分析の試み

執行健人[*1]，清光英成[*2]，大月一弘[*2]
*1 The Hong Kong University of Science and Technology
*2 神戸大学大学院国際文化学研究科

Toward Programming Learning Analysis with Code Revision

Kento Shigyo[*1], Hidenari Kiyomitsu[*2], Kazuhiro Ohtsuki[*2]
*1 The Hong Kong University of Science and Technology
*2 Graduate School of Intercultural Studies, Kobe University

本研究はより教育効果の高い指導を実現することを目的とし，プログラム作成過程で生まれるデータを活用することで学習者の学習（進捗）状況を容易に把握できるフレームワークを提供する．特に作業過程における学生のつまずきや，作業進捗が芳しくない学生の把握を試みる．プログラミング初学者を対象としたHTMLの授業において，学生のコード更新量や保存回数等のコーディング履歴を収集した．本稿では，これらのコーディング履歴を利用することで，学生の学習・進捗状況をある程度把握できるかどうかを議論する．コーディング履歴を分析して得られた学生の作業パターンや，進捗状況に関する特性ならびに，一般的なプログラミング言語の授業への応用の可能性について考察する．

キーワード: プログラミング教育，プログラミング初学者，Web，HTML

1. はじめに

近年，ICTリテラシーは現代の生活や仕事において重要なスキルとして認知され(1)，幅広い分野から関心が高まっている．コンピュータの仕組みや情報・通信技術について理解し，それらを効果的に活用するため，今や大学では情報系学部に限らず，文系学部においても，情報教育の一環としてプログラミングの授業が開講されている．学生にとってWebは日常生活で使う機会が多く，身近な題材として学習することができる．HTMLとCSS，JavaScriptなどのWeb関連技術は，プログラミング教育の導入として有用である．コンパイルを要するプログラミング言語とは異なり，エディタとブラウザがあればすぐにコードの結果を確認できる上，学習コストは比較的低い．動的なWebサイトを作ることができるようになるためJavaScriptを埋め込むことにより，高度なプログラミング学習の橋渡しとなることが期待できる．

プログラミング学習は初学者にとって容易でなく，効果的な教授法や学習方法等に関する様々な研究が行われてきた(2)．主な研究対象はCやJavaといったプログラミング言語の授業である． Web技術を扱う授業を対象とした研究では，授業中に顕在化した問題や，ある程度完成したコードに含まれるシンタックスエラーの分析等が行われた．しかし学生は何を質問すれば良いか分からないといった理由等で質問を躊躇することもあり，そのような顕在化しない問題の発見や対応は困難である．またある程度プログラムを完成させる前の作業過程における学生のつまずきや，作業進捗が芳しくない学生の把握ができれば，より効果的な指導や形成的フィードバックが期待できる．そのためには，コードの提出（コンパイル・バリデーション）時のデータだけでなく，作業過程におけるより細かい粒度のデータを分析する必要がある．

本研究は，プログラム作成時に生まれるデータを活用することで，プログラミング授業の講師が学生の学習状況を容易に把握できるようにし，より教育効果の高い指導を実現することを目的としている．特に作業

過程における学生の潜在的な問題やつまずきに対して効果的な指導を可能にするため，コードの更新量や保存回数といったコーディング履歴を分析・可視化することで,学生の進捗状況を明らかにすることを試みる．本稿では，コーディング履歴から学生の学習・進捗状況をある程度推察できるかどうかを議論する．そのためプログラミング初学者を対象とした HTML の授業における学生のコード更新量や保存頻度等のデータを分析し，得られた学生の作業パターンや，進捗状況に関する知見を報告する．また HTML の授業だけでなく，一般的なプログラミング言語の授業への応用の可能性について考察する．

2. 関連研究

Omer ら[2]は初学者を対象としたプログラミング教育に関する系統的レビューを行った．2014 年から 2020 年の期間に主要な学会誌等に掲載された 69 の文献を整理し，当該分野で主な研究対象となっている５つの項目(教授法・学習，評価，コンテンツ，ツール)を特定した． 学習に関する文献はさらに個別・協調学習に分けられるが,本研究は個別学習における学習過程・行動に焦点を当てている．

Fu ら[3]は C 言語の授業におけるプログラミング初学者の学習行動を理解することでその弱点や問題を特定することを目的とし，リアルタイム学習分析システムを開発した．サーバーに蓄積される学生のコンパイルログを収集・処理することで，コンパイルエラーやその回数，分布に関するデータ可視化をリアルタイムで提示することができる．ClassCode[4]は JavaScript の授業における学習・教授を支援する Web ベースのプログラミング教育システムである．学生は自身のペースでインタラクティブなコーディング課題に取り組むことができる．システム上で書いたコードや実行したテストケースの結果等のデータは学生ごとに収集・集約され，講師が学生の学習状況を把握できるように可視化される．

Park Susan[5]は初学者向けの Web 開発の授業において投稿された質問に対してコンテンツ分析を行い，学生が抱えた問題を教務・コンテンツ・デザイン・開発・技術・その他に分類した．加えて，開発に関する問題はさらにハイパーリンクやリスト，テーブルといった HTML の各文法に分類した．Park ら[6]はコードのバリデーション時に発生した HTML と CSS のシンタックスエラーに焦点を当て，学生がそれらをどのように解決したのかを分析した．受講者のうちほぼ全ての学生の課題になんらかの未解決のシンタックスエラーが見られ，特に入れ子構造，親子関係を必要とする文法が多くの学生にとって問題となることが分かった．シンタックスエラーの分析に焦点を当てた研究ではであるが, Web ベースのシステムにより学生のコードに関するデータが収集されるという点が本研究と類似している．

これらの研究は，学生の質問やコードからコーディング中に生じた問題やエラーの特性を明らかにすることを目的としている．本研究では，コード更新量と保存頻度といったコーディング履歴を利用することで，学生の学習・進捗状況を容易に把握できるようにする点が異なっている．本稿では，学生の学習・進捗状況の分析を目的としたコーディング履歴の有用性について議論する．第３章でデータ収集について述べ，第４章でコード更新量・頻度とコードを分析することで得られた学生の作業パターン, 進捗状況に関して述べる．第５章で考察を論じ，第６章はまとめである．

3. データ収集

プログラミング初学者を対象とした HTML の授業において, 各学生が課題への解答として作成した HTML ファイルを保存（更新）毎に収集した．表１に授業で扱われた主なトピックを週ごとに示している．従来は対面形式で授業が行われていたが，昨年度から COVID-19 による感染拡大防止のため，オンデマンド形式で行われていた． 教材は講義動画，テキスト，テンプレートコードがあり，質問の機会はオンライン上で与えられた．２クォーターに渡る授業において履修者 137 名，140 名のうち，124 名，103 名のデータを学生の同意を得た上で収集した．大半の学生が両クォーターの授業を続けて履修した．Atom をテキストエディタとして使用し，データ収集を行う拡張機能を開

表 1: 授業の主なトピック

週	トピック
第 1 クォーター（124 名）	
1	授業概要，環境構築
2	Web ページ制作，HTML,テキスト
3	画像，CSS によるデザイン
4	ハイパーリンク，class
5	ボックスモデル，id
6	課題（Web ページレポート作成）
7	スタイルシート，ボックスレイアウト
第 2 クォーター（103 名）	
1	授業概要(続)，環境構築
2	テーブル，テーブルレイアウト，色
3	テーブルレイアウト(続)
4	テーブルレイアウト(続)
5	テーブルレイアウト(続)
6	HTML5, canvas, JavaScript による描画
7	JavaScript によるアニメーション

発した．学生が HTML ファイルを保存する度にそのコード, 保存時刻, 学生の唯一の識別子（以降学生 ID）及び授業の識別子をサーバーに送信する．サーバー側ではそれらのデータを受信し，データベースに保存する．サーバーと通信できない場合も想定し，収集データを一時ファイルに保存，通信回復時にサーバーに再送信する機能を備えている．各クォーターで収集したデータの総レコード数は 12,070, 16,484 である．学生のレコード数の平均・標準偏差は(98.0, 92.8)，(160.0, 116.1)である．これらは学生がファイルを保存する頻度を表している．

収集したデータを確認するための Web アプリケーションを開発した．図 1 は第 2 クォーターの第 4 週目のある学生のデータの詳細画面を示している（学生 ID 等の個人情報は伏せている）．インターフェースとして 3 つのコンポーネント（学生情報，コード，プレビュー）が画面左，中央，右に配置されている．学生情報コンポーネントでは学生 ID と作成されたファイルのリストが表示される．コードコンポーネントでは学生が書いたコードが表示され，上部に配置された矢印ボタンによって表示するデータを保存時刻の順に変更することができる．プレビューコンポーネントには，当該の HTML ファイルを Web ブラウザで開いた際に描画さ

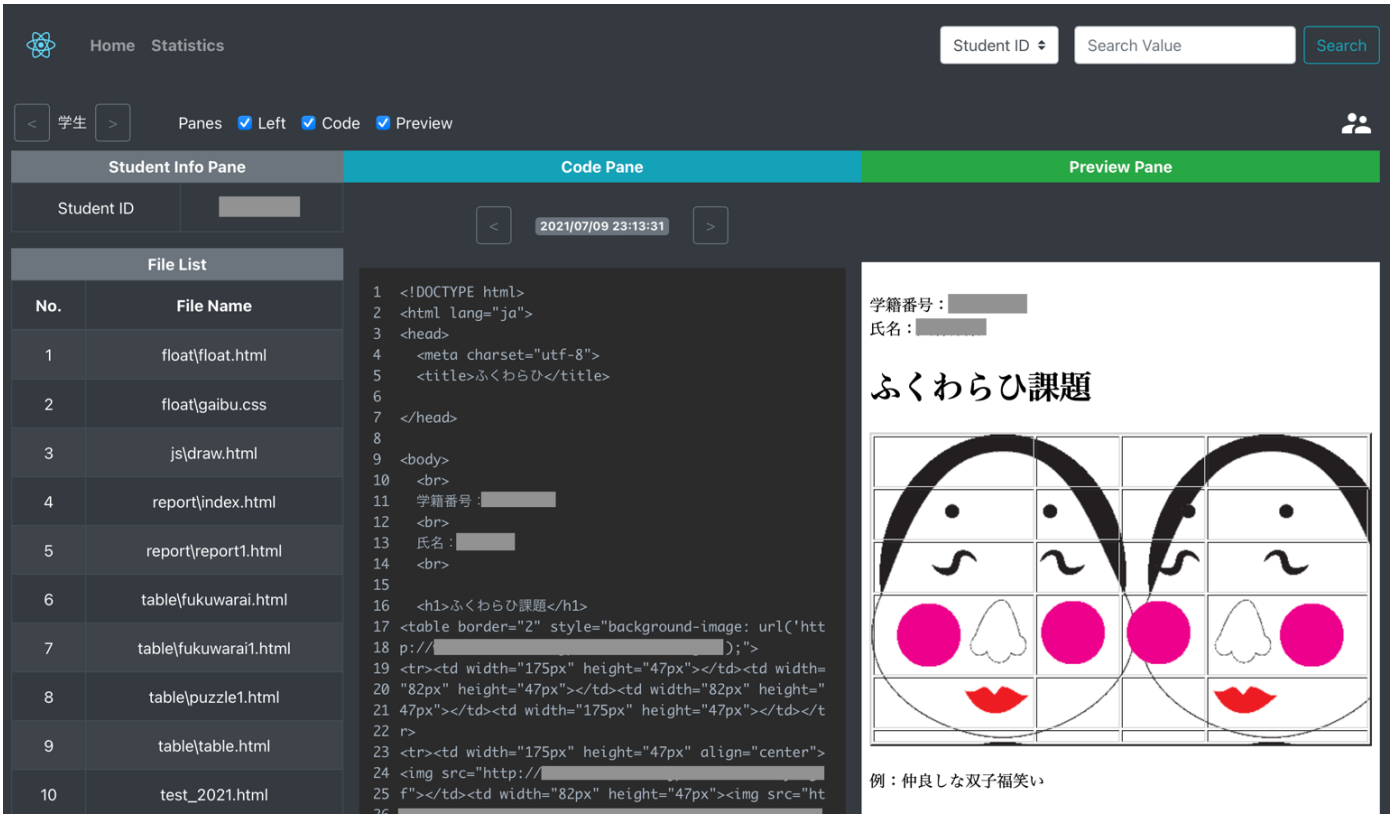

図 1: 第 2 クォーター第 4 週目のある学生のデータの詳細画面

れる画面が表示される.

4. コーディング履歴の分析

本章では，学生のコーディング履歴をコードの更新量・保存間隔の観点から分析して得られた学生の作業パターンについて述べる. 本稿ではコードの更新量を，あるファイルが保存されてから次に保存されるまでに追加または削除された文字数とする．あるファイルに追加された文字数が削除された文字数より大きい場合，コードの更新量は正の値となり，そうでない場合は負の値となる．あるファイルが作成されて初めて保存された際には，コードの更新量をその時点のファイルの文字数とする．図 2 は，全ての学生の各レコードのコード更新量を，課題ごとに示している．コード更新量を縦軸にとり，学生 ID を横軸に並べている．また図 3 にコード更新量の平均と標準偏差を課題ごとに示している．コードの更新量に関して 1) 顕著な正の更新量 2) 顕著な負の更新量 3) 長時間に渡る頻繁な更新の 3 つに注目し，分析した.

4.1 顕著な正の更新量

正の更新量が多い上位 30 件のレコードに含まれるコードを調べ，学生がコードを大量に追加し，保存した際の行動パターンを 4 つに分類した.

- パターン 1 ：コードの複製（一連のコードを繰り返し貼り付ける）
- パターン 2 : 完成コードのみ保存（ファイルの新規作成後の作業過程なし ）
- パターン 3 : テンプレートコードへ一連のコードを追加（HTML コンテンツの追加）
- パターン 4 : コードの復旧（一連のコード削除後に元に戻す）

パターン 1 では，ある一連のコードを複数回コピーすることで, HTML コンテンツを生成するレコードに見られた．例として，最も大きい正の更新量(5467 文字)を含むレコードでは, 多くのテーブルセルから成るテーブルが作成されていた. 直前のレコードにおいて，当該学生は数個のテーブルセルから成るテーブルを作成し，保存している．その後テーブルセルを大量にコピー・保存することで，顕著な正の更新量となっている．HTML には for ループの様な命令を一定回数繰り返す文法がないため，同一のタグを大量に生成するに

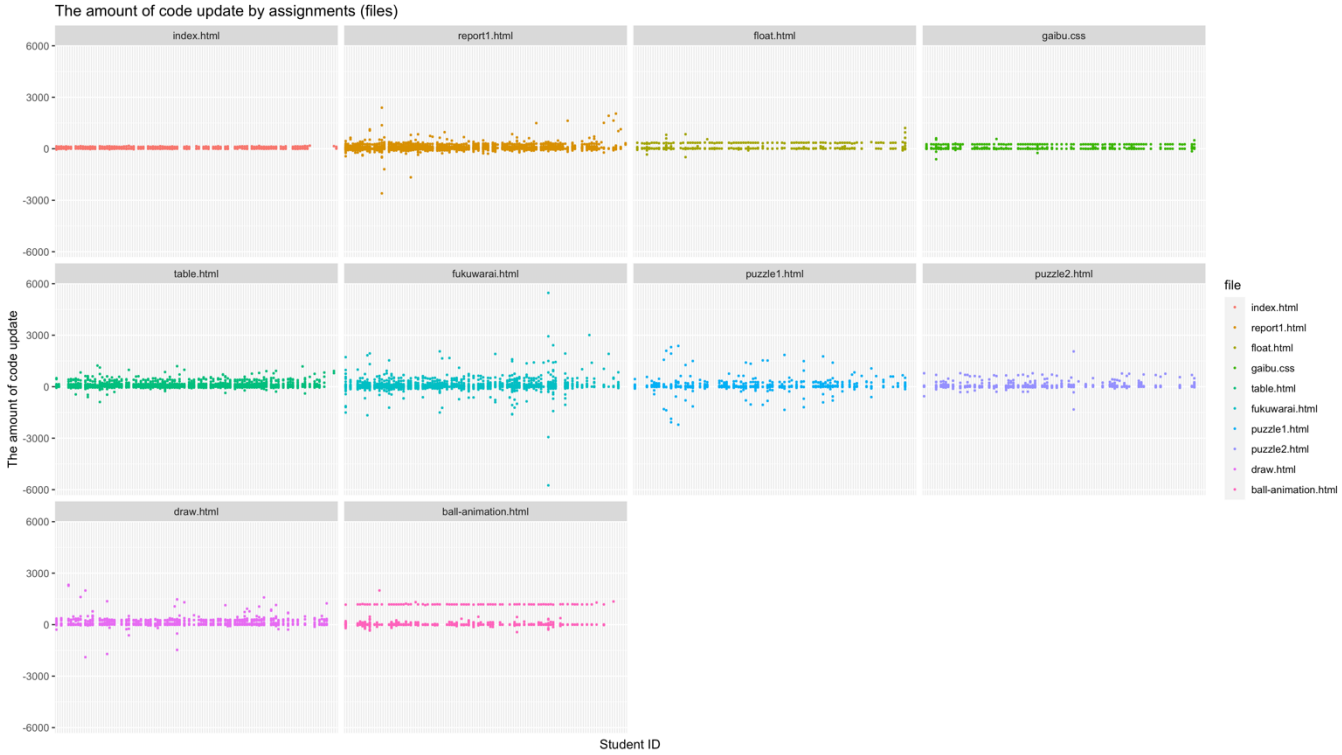

図 2：全学生の課題ごとのコード更新量

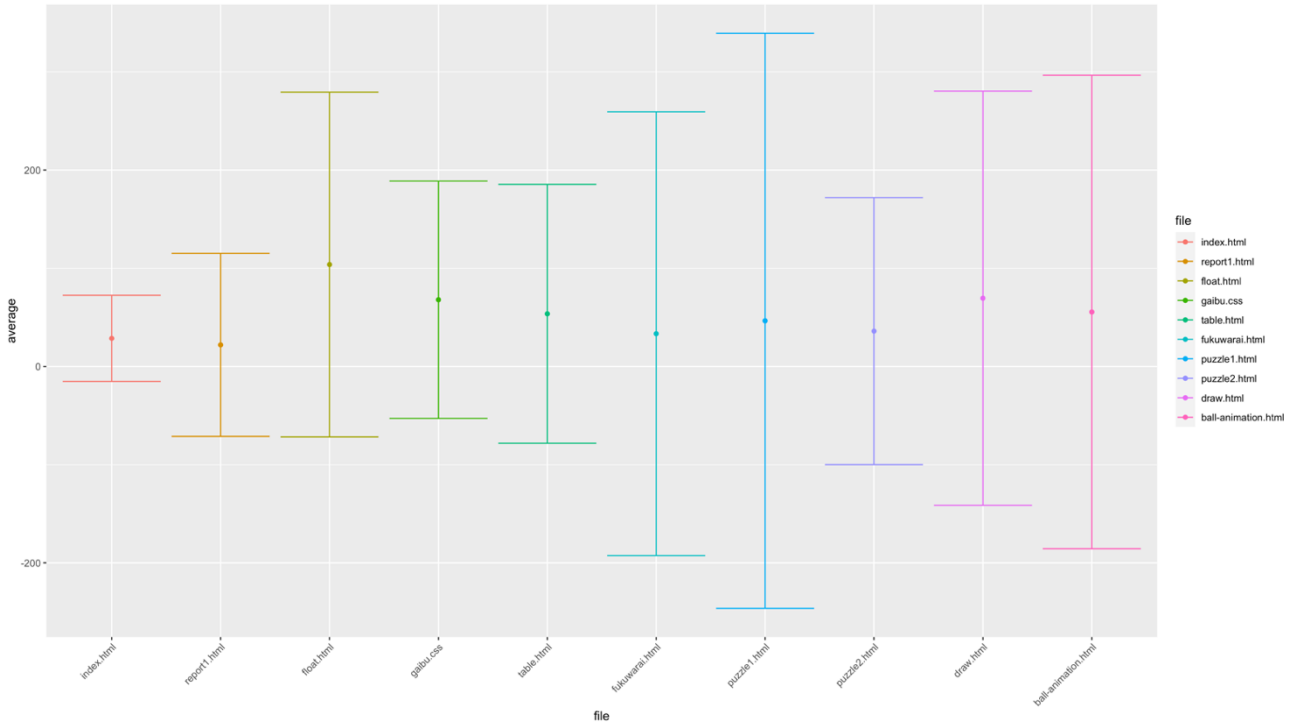

図 3：全学生の課題ごとのコード更新量の平均と標準偏差

は，手動でコピーを繰り返す必要がある．この様な作業パターンは HTML に特有であり，学生のレコードに見られた場合，当該学生は HTML の構造をある程度理解していると考えることができる．

パターン 2 では，ある課題を完成するための作業ログが一切なく，完成したコードのみがレコードとして残っていた．この場合，当該学生は別の学生から完成した課題のコードを譲り受け，そのままエディタ上に貼り付けて保存したことが考えられるため，講師は注意が必要である．

パターン 3 では，テンプレートとなる HTML のコードが作成された後，大量の HTML コンテンツが追加されていた．例として，ある学生はテーブルを含むある課題のコードを別の課題に再利用するため，テーブルセルの中身だけを削除したのち，必要なコンテンツを大量に追加していた．またある学生はレポート課題の Web ページ制作のため，まずレポートの枠組みを HTML と CSS で作成したのち，大量の文章を一括して追加していた．

パターン 4 では，ある HTML コンテンツのコードが一時的に削除され，直後に元の状態に戻されていた．例として，ある学生は作成した 2 つのテーブルのコードを削除して保存した後，次の保存時にはそれらのテーブルのコードを戻していた．当授業において，HTML のコメントについては教授されていない．この様な作業は，HTML の構造を確認することを意図としたコメントアウトに当たると考えられる．

4.2 顕著な負の更新量

負の更新量が多い上位 30 件のレコードに含まれるコードを調べ，学生がコードを大量に削除し，保存した際の行動パターンを 3 つに分類した．

- パターン 1 ：コードの書き直し
- パターン 2 ：HTML コンテンツの削除
- パターン 3：テンプレートの生成（類似ファイルから不要な部分を削除）

パターン 1 では，学生はある程度コードを完成させていたにも関わらず，後で同じコードを初めから書き直していた．この様な作業の理由として，作業ファイルを誤って削除したため，コードを書き直す必要があったことが考えられる．

パターン 2 では，ある HTML コンテンツのコードが部分的に大量に削除されていた．例として，ある学生は作成したテーブルのセル数を半減させていた．また別の学生は，作成した HTML のボタンとそれらに

対応する JavaScript のコードを半数程度まで削除していた．顕著な正の更新量における第一のパターンと同様に，この様な作業が見られる学生は HTML の構造を理解していると考えられる．

パターン３では，ある課題のファイルをコピーし，不要なコンテンツを大量に削除するとで，別の似た課題のためのテンプレートとしていた．例として，ある学生は複数の画像をテーブルセルに適切に配置する課題ファイル(fukuwarai)をコピーし，テーブルタグを削除した．その後，colspan や rowspan 等の属性を利用してテーブルセルの連結を学ぶための課題(puzzle)で必要となる新しいテーブルを作成した．

4.3 長時間に渡る頻繁な更新

プログラムにおける何らかのエラーを解決しようとする際には，問題箇所の特定や動作確認のために，一部のコードを削除・追加することがある．エラーをすぐに解決できず，学生がつまずいている場合には，長時間に渡り頻繁なコード更新量が続くと考えられる．このような仮説の元，長時間に渡り頻繁に小さいコード更新量が続いた学生を抽出するため，図 4 のようなコードの保存時刻と更新量を散布図として可視化するプログラムを作成した．学生，作業日ごとに横軸に保存時刻，縦軸にコードの更新量をとり，マーカーをファイル名ごとに色分けしている．目視での確認により 13 名の学生を抽出し，当該のコードを確認した結果，２つの作業パターンが見られた．

- パターン１：エラー解決の試み
- パターン２：Web ページの体裁の微調整

パターン１では，仮説通りエラーの解決が試みられていた．しかし大半の場合はパターン２の作業が見られ，主に画像やテーブル等の HTML コンテンツの大きさや配置，色といった Web ページの体裁に関わる微調整が行われていた．

例として図 4 に，ある学生のコーディング履歴を散布図として可視化した．5 月 28 日（課題の締め切り当日）の午後 9 時から午後１１時頃，また 8 月 8 日（課題の締め切り 2 日前）の午前 1 時から午前 5 時頃にかけて継続的なコードの更新が見られる．5 月 28 日にはエッセイを含むレポートを Web ページとして作成する課題に取り組んでおり，作業開始時点で文章は完成させていた．当日は主にレポートの体裁を整える作業をしており，画像やハイパーリンクの追加や，body・div 要素の背景色の変更などが行われていた．8 月 8 日には JavaScript を利用した図形のアニメーション課題に取り組んでおり，午前 1 時の作業開始時点ではテンプレートコードを保存したためコード更新量が 1000 を超える記録が見られる．その後表示させる図形の大きさや速度を適切に変更することができたが，canvas 要素に画像を表示させる際に文法エラーが発生した．10 分ほど文法エラーを解決しようとするコードの更新が見られたが解決せず，午前 3 時半ば頃に新規作成したファイルを用いて画像を表示するコードを

図 4：ある学生のコードの保存時刻・更新量の例

書いた．その後元のファイルにコードを追加することで適切に画像を表示させ，その後図形の色や位置・速度の微調整が行われていた．

5. 考察

コードの更新量・保存頻度といったコーディング履歴の特徴に注目することで，学生の作業パターンをある程度分類することができた．これにより，講師は学生のコードを見なくとも学習・進捗状況を把握でき，より教育効果の高い指導が可能になると期待できる．前章で述べた様に，実際にコードを調べて分析した限りでは，大抵の場合学生は順調に作業を進められていることが分かった．しかし，顕著な正のコード更新量のパターン 2（完成コードのみ保存），顕著な負のコード更新量のパターン 1（コードの書き直し），長時間に渡る頻繁な更新のパターン 1（エラー解決の試み）で見られたように，教師の対応やアドバイスが必要だと考えられる作業パターンも散見された．講師がこの様な作業パターンを特定しようとする際には，図 4 に示したコードの更新量と保存時刻の散布図に加えて，それぞれ 1）作業ファイルごとの保存回数 2）同一作業ファイルの文字数の変化　3）シンタックスエラーの継続時間といったデータが有用になると考えられる．特に継続的な小さいコード更新量が見られる時にあるファイルの別名ファイルを作成して作業している場合，学生は何らかのエラーを解決しようとしていると考えられるので，ヒントやアドバイスを与える適切なタイミングとなる可能性がある．

また HTML の文法上，HTML に特有の作業パターンは，顕著な正の更新量のパターン 1・パターン 3，顕著な負の更新量のパターン 2・パターン 3 である．これらの作業パターンを検出することで，コードを逐一見なくても学生が HTML の文法や構造を理解しているかどうか判断できる可能性がある．それ以外の作業パターンは，HTML の授業だけでなく，一般的なプログラミング言語の授業においても見られると考えられる．コードの更新量から学生の学習状況を把握するための観点として応用が期待できる．

6. まとめ

本稿では，コードの更新量・保存間隔といったコーディング履歴から，学生の学習・進捗状況をある程度推察することが可能であることを議論した．プログラミング初学者を対象とした HTML の授業において収集した学生のコーディング履歴を利用し，1）顕著な正の更新量 2）顕著な負の更新量 3）長時間に渡る頻繁な更新が見られた際の学生の作業パターンを，実際に学生が書いたコードを例に挙げて論じた．特に講師が注意を要する学生の作業パターンとして，顕著な正のコード更新量から作業記録のない完成したコードの検出，顕著な負のコード更新量からコードの一からの書き直し，長時間に渡る頻繁な更新からエラー解決の試みが推察しうることを示した．これらのコーディング履歴の分析・可視化により，講師は学生のコードを見なくとも学習・進捗状況を把握することができ，より教育効果の高い指導が可能になると期待できる．

今後の展望として，収集した学生の HTML コードのバリデーションを行い，シンタックスエラーとコードの更新量・保存間隔について分析を進める予定である．また学生・課題ごとにコードの更新量・保存間隔の変化を分析する．今回データ収集を行なった授業はオンデマンド形式で実施されたため，学生の学習時刻が不定であった．学生の学習状況を比較・分析するため，学習時刻に関して何らかの処理が必要である．その上で，通常とは異なる作業パターンを検出することで問題の発生を特定する手法等の開発を検討している．

謝辞

本研究の一部は科研費(19K03000)「プログラミング教育のための進捗把握手法」の支援による．ここに記して謝意を表す．

参 考 文 献

(1) Trilling, Bernie, and Charles Fadel. 21st century skills: Learning for life in our times. John Wiley & Sons, 2009.

(2) Omer, Uzma, Muhammad Shoaib Farooq, and Adnan

Abid. "Introductory programming course: review and future implications." PeerJ Computer Science 7 (2021): e647.

(3) Fu, Xinyu, et al. "Real-time learning analytics for c programming language courses." Proceedings of the Seventh International Learning Analytics & Knowledge Conference. 2017.

(4) Suzuki, Ryo, Jun Kato, and Koji Yatani. "ClassCode: an interactive teaching and learning environment for programming education in classrooms." arXiv preprint arXiv:2001.08194 (2020).

(5) Park, Thomas H., and Susan Wiedenbeck. "Learning web development: Challenges at an earlier stage of computing education." Proceedings of the seventh international workshop on Computing education research. 2011.

(6) Park, Thomas H., Brian Dorn, and Andrea Forte. "An analysis of HTML and CSS syntax errors in a web development course." ACM Transactions on Computing Education (TOCE) 15.1 (2015): 1-21.