5H-01

授業支援のためのビジュアルプログラミングの編集履歴に基づく学習分析

中澤 真†
会津大学†
短期大学部

梅澤 克之‡
湘南工科大学
工学部

1. はじめに

プログラミングの授業支援システムの研究のほとんどが，テキスト型プログラミング言語を対象としたものであり，分析に利用する特徴量はコンパイルやプログラムの実行といった操作履歴と，コンパイル時の文法エラーとなっている[1][2]. このため，Scratch のような文法エラーが生じないビジュアルプログラミング言語に対応した授業支援システムはまだない．本研究では Scratch の編集履歴を学習分析することで，授業改善に必要な情報を教員へフィードバックする方法について論ずる．

2. 学習分析と編集履歴可視化システム

2020 年度から全面実施となる小学校でのプログラミング教育に向けて，2019 年度までの移行期間で先行的に授業を展開している学校では，代表的なビジュアルプログラミング言語である Scratch が利用されることが多い．Scratch は初学者でもプログラミングを容易に学ぶことができるように設計されてはいるが，小学生が授業として学ぶ場合には，やはりサポートが不可欠なものとなる．我々が地域の小学校と連携して取り組んできたプログラミングの授業においても，2 人以上の TT (Team Teaching)体制が授業運営には欠かせないことが明らかになった．プログラミングの作成にあたっては多種多様な誤りやつまずきがあるため，それぞれの学習者の進捗や理解状況を教員が授業中に把握するには時間をかけて巡回指導する必要があることが大きく影響している．この指導の際にも，プログラムの誤りの原因を発見するのに時間を要してしまうこともよくある．一般的なテキスト型プログラミング言語と異なり，Scratch のようなビジュアルプログラミング言語では文法エラーは基本的に発生しないため，発見すべきは論理エラーのみとなるため対応に時間がかかることになる．サポートの教員が確保できれば大きな問題にはならないが，大規模校でも通常授業では 2 人の TT 体制が限界であるため，小・中規模校ではサポート無しとなってしまう場合が多い．このため，授業実施時間中に，学習につまずいていたり，他の学習者と異なるタイプのプログラムを作成している者など，教員がリアルタイムに状況把握できる授業支援システムが求められることになる．

これを実現するためには，学習者が授業中に作成するプログラムの編集履歴を活用する必要がある．筆者らはプログラムの編集履歴可視化システムを既に構築している[3][4]. 図 1 に示したように，このシステムでは，10 秒ごとに学習者が編集中のプログラムをサーバ上に自動的に保存し，記録された編集履歴を学習者別，タイムライン別に可視化できるようにしている(図 2 参照)．

しかし，この可視化機構は授業終了後にバッチ処理で学習者のモデリングや理解度・学習効果などを検証して授業改善に役立てることは可能だが，授業中のクラス全体の傾向や逸脱しいている学習者を短時間で把握することには適していない．そこで本研究では，リアルタイムな状況把握のために，プログラムの編集履歴からどのような特徴量を抽出すればよいかについて論じる．

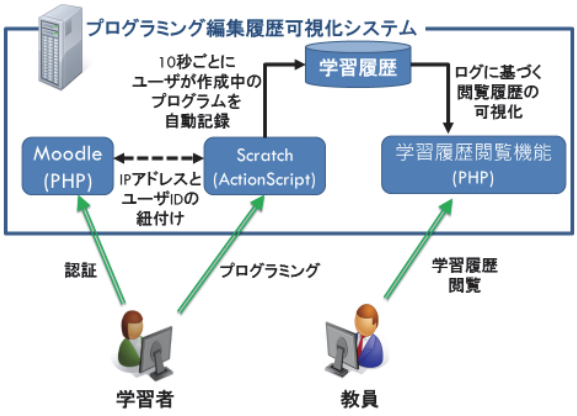

図 1:編集履歴可視化システム構成図

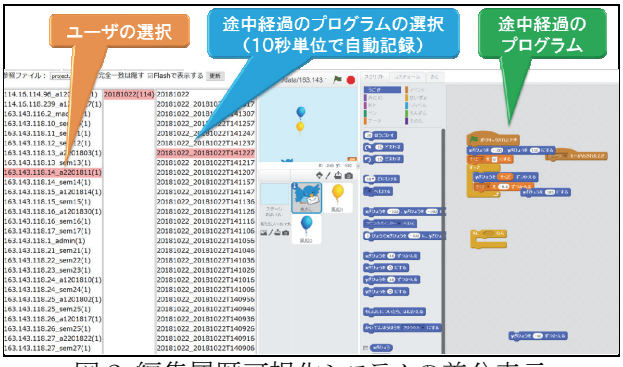

図 2:編集履歴可視化システムの差分表示

3. クラス全体の傾向把握のための学習分析

Scratch のバージョン 2.0 では，作成されるプログラムは JSON 形式で保存されている．これを解析する際に，構文解析のような大量のバックトラッキングを必要とせずに，リアルタイムの授業支援ができることが望ましい．そこで， Scratch のプログラム中に使用されているブロック

Learning Analytics based on Edit Histories of Visual Programing for Teaching Support
†Makoto Nakazawa, University of Aizu
‡Katsuyuki Umezawa, Shonan Institute of Technology

について，その種類ごとにクラス中での使用率を特徴量とすることを考える．学習者が特定の課題のプログラム作成を目指す場合，使用するブロックには偏りが生じるため，使用率の推移を把握することによってクラス全体の進捗状況を把握することが可能になる．図 3 の例では，作業開始の 60 秒後に手の早い学習者と，まだ迷っている学習者とに差が表れているが，180 秒後には「緑の旗をクリックしたとき」というイベントハンドラとなるブロックは全員が設置済みであることが確認できる．さらに，300 秒後には，「左に回転」ブロックを設置するところまで半数の学習者が到達しており，一方誤ったブロック「右に回転」を使ってしまっている学習者がわずかにいることも確認できる．

もちろん，課題に対するプログラムは一つとは限らず，図 4 に示したように学習者ごとに異なるブロックを用いて課題のプログラムを実現している場合もある．しかし，初学者が取り組むプログラムの場合には，欠くことのできないブロックが必ずあるため，この特徴量でクラス全体の傾向は十分把握可能である．

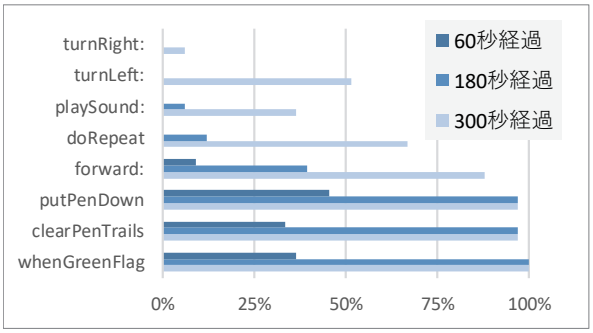

図 3:プログラムブロック別の使用率の推移

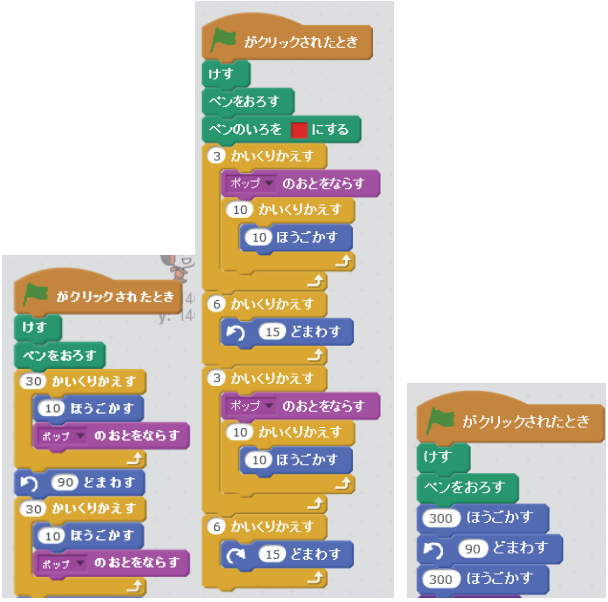

図 4:学習者によるプログラムのばらつき

もう一つの特徴量は制御ブロックのネスト構造の使用割合である．ネスト構造は初学者の典型的なつまずきポイントであるため，どのように入れ子を使い，またどの程度の学習者が適切にネストを完成させているかを把握することが授業運営には欠かせない．図 5 に示した例では，繰り返しの中に条件分岐を組み込むことが時間経過とともに進み，対して条件分岐どうしのネストは進捗が遅れ気味であることがわかる．

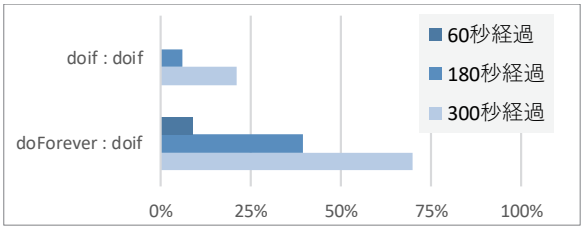

図 5:ネスト構造の使用率の推移

これらの特徴量はクラス全体の傾向把握以外に，逸脱したプログラムを作成している学習者を抽出することにも活用可能である．逸脱は誤ったプログラムというだけでなく，斬新なアイデアで作成されているケースもあるため，そういったプログラムを効率よく抽出し紹介できるようにすることは授業運営において有用である．

4. まとめと今後の課題

本研究では，ビジュアルプログラミング学習のためのリアルタイム授業支援を実現するために，学習分析に必要な特徴量について示した．これらの情報を逐次提示することにより，教員はクラス全体の進捗や，誤ったプログラムブロックの使われ方の傾向，さらには逸脱したプログラムを効率よく把握することが可能になる．

今回は紙面の関係で割愛したが，ブロックの順序を考慮した使用率や，条件式，演算の使用率も有用な特徴量となりえる．また，今回の学習分析はある時点における学習履歴を学習者間で比較することで実現しているが，学習者別の履歴を時系列的に分析することで，ブロックの削除の繰り返しや，プログラムが変化しない時間などを特徴量とすることで，学習者の状態を「放棄」「混乱」というように判断することもできるようになる．

今後，これらの特徴量を現場の教員が把握しやすいインターフェースとなるように実装を進めていく予定である．

謝辞

本研究の実施にあたり会津若松市立一箕小学校大橋淳子前校長はじめ諸先生方には多大なるご協力をいただいた．本研究の一部は独立行政法人日本学術振興会学術研究助成基金助成金基盤研究(C) 17K01101の助成による．

参考文献

[1] 加藤，石川，“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)，pp. 1918-1930，2014.
[2] 星野裕樹ら，“プログラミングにおけるコーディングスタイルの学習：コード記述の特徴解析手法の提案，”信学技報 ET,115(352)，pp.31-36，2015.
[3] 中澤真，荒本道隆，後藤正幸，平澤茂一，“編集履歴可視化システムを用いたLearning Analytics ～ Scratchを用いた初等教育向けプログラミング教育における学習者の思考パターン分析，” 情報処理学会第78回全国大会講演論文集，pp. 4-531-4-532，2016.
[4] 中澤真，梅澤克之，後藤正幸，平澤茂一，“Scratchを用いたプログラミング学習時の閲覧履歴・編集履歴・脳波履歴を組み合わせた学習者分析，” 情報処理学会研究報告，CE-138 No.1,pp.1-6，2017.