N-007

プログラミング初学者教育における教員補助を目的としたログデータの解析および可視化方法の提案

Log-Data Analysis and Visualization for Beginner Programming Class as Teacher Support Tool

橋本玄基 † Genki Hashimoto
大枝真一 ‡ Shinichi Oeda

1. まえがき

日本の学校教育は一対多形式がほとんどであり，学生の理解度などを授業毎に把握することは難しい．しかし，実習を伴うプログラミングの授業では学生の行動を確認できる UNIX コマンドヒストリや，ソースコードの編集履歴などのログデータを容易かつ自動的に保存することが可能である．一方で，世界的に Massive Open Online Courses(MOOCs) と呼ばれるオンラインでの大規模教育サービスが行われており，そこから取得可能な多量のログデータから，学生をモデル化する教育データマイニングが世界的に流行している． 学校での授業と MOOCs の間には受講人数に大きな差があるため，授業改善のための分析には MOOCs とは違ったアプローチが必要となることは明白である．そこで，本研究では対象を MOOCs と比べると規模が小さい学校教育のプログラミングの授業にしぼり，授業改善や教員補助，具体的には要支援者を予測し早期に発見するためのログデータを活用するためにデータ解析や効果的な可視化方法について検討する．

2. 先行研究と本研究の独自性

授業改善を目的としたプログラミング授業のログデータの取得・活用を行っている研究は日本において多く行われている．その多くはより具体的な情報を得るため，独自のコーディングツールや総合開発環境を作成し，そのツール上で学生にコーディングをさせることで，ログデータを取得・解析している [1–4]．しかし，初学者を対象とした教育においては，プログラミング言語の知識やアルゴリズムだけではなく，コンパイルのオプションや一般的なエディタの使用方法なども含めて講義する必要があり，専用のツールでコーディングを行うことは適切ではない．そこで，本研究では専用のツールを用いず，一般的な情報技術教育に用いられる UNIX 環境下でのプログラミング学習を対象とし，UNIX 既存の機能やシェルスクリプトを用いて取得可能なログデータから授業支援に有効な情報を得ることを目指す．

スキル推定や支援者予測のためのクラス分類などには，一般的にはラベルのついたデータ群を用意し，それらを教師あり学習手法を用いて評価器に学習させることで未知のデータに対する予測を行うことが多い．しかし，学校の授業においては，データが少ないため過学習の恐れがある．また，MOOCs などと異なり，授業形態は教員や年度によって大きく変化する．そこで，我々は授業毎における行動の違いを教師なし学習によって分類することを考えてきた．これまでの研究においては，コマンドヒストリから特徴ベクトルを作成し，非階層クラスタリングを用いて学生の傾向をつかもうとしてきた [5–8]．データの特性や量，および解析の結果から，スキル推定は困難であると考え [6]，要支援者の予測，特に作業量が多く授業に集中しているが質問が少なく，つまづきを解決できていない学生を抽出することを目的としている [7, 8]．具体的にはクラスタリングを行い，少数クラスタを特殊な学生とし，少数クラスタには授業ペースよりも早く進めているか，遅れている学生のどちらかが属しているという仮定のもと解析を行っている．本研究も最終的な目標は要支援者の予測である．

本研究では，別のアプローチとして，教員の知見を活かし，判断を任せる方法を考える．具体的にはソースコードが授業中どのように増減するかをリアルタイムで可視化するシステムを構築し，教員はそれを見ながら通常通り授業を行う．可視化により学生へのケアがしやすくなったか，欲しい情報が得られているかを調査する．

3. 解析対象

プログラミング初学者を対象に行われる C 言語の学習のための 90 分の授業および試験のログデータを使用する．現在取得可能なログデータは発行した UNIX コマンドの履歴とコンパイルの対象となったソースコードファイル，コンパイルした時刻の対象ディレクトリのサイズ (バイト数) である．

4. ソースコード変量可視化システム概要

本研究で作成したソースコード変量可視化システムは gcc 埋め込み部，可視化プログラムの 2 つからなる．概要図を図 1 に示す．また，学生には授業中は指定のディレクトリ下で作業をさせる．

gcc 埋め込み部は従来の gcc コマンドを binary code を置き換える，または shell の設定ファイル上で gcc に alias をかけることで自作の shell スクリプトに置き換える．その後以下の順番で処理を行う．

† 木更津工業高等専門学校 制御・情報システム工学専攻, Advanced Curse of Control and Information Engineering, National Institute of Technology, Kisarazu College
‡ 木更津工業高等専門学校 情報工学科, Department of Information and Computer Engineering, National Institute of Technology, Kisarazu College

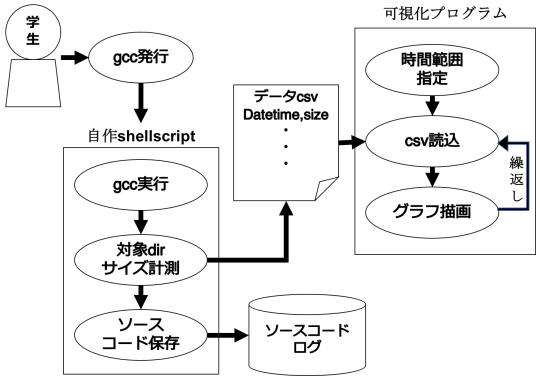

図 1 システム概要図

1. 引数を全て渡して，従来の gcc コマンドを通常実行．
2. 指定した作業ディレクトリ内の.c,.h ファイルの合計サイズを測定．
3. コンパイル時刻，および取得した合計ファイルサイズを csv に追記．
4. コンパイルに使用したソースコードをログ保存領域にコピー．

これにより，与える制限は作業するディレクトリの場所のみで学生は今まで通りエディタなどの制限なく授業のプログラムを作成でき，さらにログデータを取得可能になっている．

可視化プログラムは授業中に実行することで gcc 埋め込み部により作られるコンパイル時刻と合計ファイルサイズを 1 行とした csv ファイルを読み込み，リアルタイムで教員が学生のコーディング量を確認することができる．

1. 描画したい日時の範囲 (授業の時間) を指定する．
2. 各学生のファイルサイズを保存した csv を読み込む．
3. 指定範囲のファイルサイズから範囲外で，開始時間に最も近いデータのファイルサイズを引く．
4. 指定範囲のグラフを描画する．グラフは各学生 1 枚ずつと，全学生のグラフを 1 枚に集めたものを描画する．
5. 定時間毎にグラフの描画を更新する．

このプログラムにより，コーディングの過程が簡易的ながらも瞬間的に見ることができ，学生同士の違いも一目でわかるため，異なる行動を行っている学生の把握が容易となる．また，グラフの情報から即座に TA が向かうなどして学生への補助が期待できる．日時の指定ができるため，過去のグラフも作成でき，ソースコードログと合わせて気になる点で具体的にどのようなコーディングを行っていたかを詳細に確認することも可能である．

5. 実験内容

本研究では，ある授業において 1 回の試行を行った．授業の開始とともに，可視化プログラムを実行し，指導教員は常にそれを確認しつつ，授業を行い，単純なシステムの使いやすさおよび実用性を調査する．また，授業後に作成されたグラフとソースコードログ，コマンドヒストリを照らし合わせ，実際に出来上がったグラフと行動との関連性を調査する．

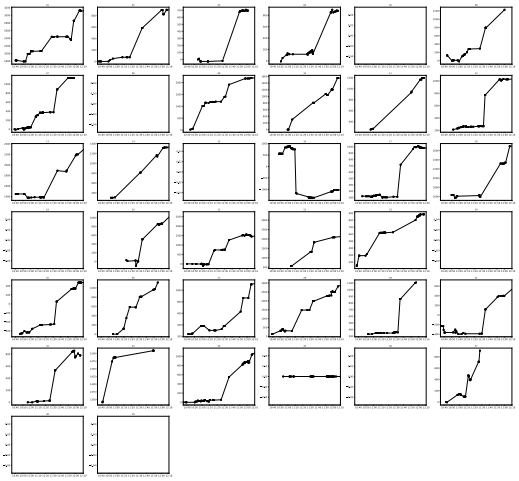

図 2 全学生のソース変量グラフ

6. 結果

6.1. システムの実行結果

全学生の可視化グラフを図 2 に示す．横軸が時刻，縦軸がシステムが取得した指定ディレクトリのファイルサイズ (単位は byte) から指定時間の前までのサイズを引いたサイズである．グラフの表示がない学生と作業をしているのにグラフが変動しない学生がいた．調査の結果，これらの学生は指定したディレクトリが存在していない，または指定したディレクトリ内で作業を行っていなかった．これらの学生を除くと，学生がどれぐらいの頻度でコンパイルを行い，その度どの程度ソースコードを書いているのかが可視化できていることがわかる．図 2 は授業終了後のものだが，授業中はリアルタイムにグラフが更新されるため，特に他の学生よりもサイズが大きくなるのが早い学生は視覚的にわかりやすい．グラフの変動には，大きく分けて急激な増加，ある程度の時間があいた増加，変動の少ない高頻度な間隔でのコンパイル，急激な減少が見られた．また多くの学生が授業終了付近までコンパイルを行っているのに対し，かなり前からコンパイルが止まっているように見える学生が 1 名いた．

6.2. システムの使いやすさの評価

実際にグラフを見ながら授業の進行を行った教員にヒアリングを行った．授業中に誰が取り組んでいて，誰が取り組んでいないかが，ログの観点から把握できる点が非常に有用であると述べており，特に授業の演習時間は学生の机の周りを歩いて回る際に，手が動いていない学生には声を書けやすいが，一生懸命コーディングしている学生は，うまくいっているように見えてしまう．その

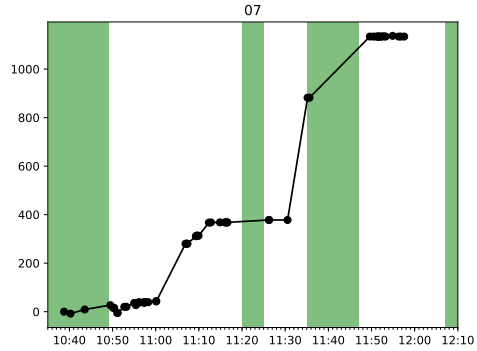

図 3 便宜番号 7 番の学生のソースコード変量

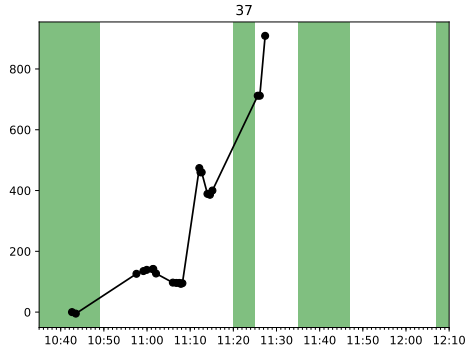

図 4 便宜番号 37 番の学生のソースコード変量

ような場合にログデータを確認することで集中してプログラムに取り組んでいる学生がつまづきの状態であることを予測し教員から声をかけられる点が評価が高かった．一方，要望として，現状教員が全学生のグラフ表示をみて，教員の判断によって注意が必要な学生を識別しているものを自動的に教えてくれるような機能や学籍番号順に画面に表示されているものを着席している位置で表示されると，さらに対応しやすくなるなどの意見があった．

6.3. グラフの意味の確認

6.3.1. 急激な増加

例として 1 人の学生のグラフを図 3 に示す．図中の横軸 11:32 付近から 11:35 付近が該当する．ソースコードの量が急激に増えることは，サンプルプログラムや今までのプログラムをコピーし動作確認のためにコンパイルを行ったことが推測される．今回の授業では 11:25 付近で新しい課題の説明プリントが配られたため，その付近で急激な増加が起き，コマンドヒストリと照らし合わせることでコピー作業によるものであることが確認された．これが周りよりも遅れている場合は別のことへの集中が疑われるため，講義への集中を促す指標の 1 つとして期待できる．

6.3.2. ある程度の時間があいた増加

ある程度の時間があいているものは，新規プログラムの作成を行ったか，教員の説明があったことが推測される．後者を確認するために，授業中に記録した教員の説明中の時間を塗りつぶしたグラフを代表として 1 名分，図 3 に示す．図中では 11:01 付近から 11:06 付近が前者に，11:35 から 11:48 付近が後者に該当していた．教員が説明中は基本的に学生の PC 画面を強制的に切る，もしくは教員の PC 画面を共有するため作業は行っていない．そのため，この 2 つは大きく意味合いが異なり，可視化の際にもわかりやすく区別することが求められる．

6.3.3. 変動の少ない高頻度のコンパイル

初歩的なプログラムでは変数の値を直接書き換えることで動作確認をするものが多く，動作確認の作業がこの傾向に当てはまると推測できる．また，バグの修正も当てはまるであろう．コマンドヒストリとソースコードログを照らし合わせた結果推測通り，2 つのパターンが存在した．どちらの場合も，特にバグの修正であった場合，長い時間続くようであれば，つまづいている状態であるため，ケアに向かうことが求められる．明確にどの程度の繰り返しのコンパイルを行うとつまづいている場合が多いかなどを検証することが求められる．さらに，コマンドヒストリと組み合わせることでコンパイルと実行が交互に繰り返されれば，動作確認かコンパイルは通るバグの調査，実行を挟んでいなければコンパイルエラーによるつまづきというようにより原因を具体的に可視化することが期待できる．

6.3.4. 急激な減少

急激な減少はサンプルプログラムを実行するために自分の作業ディレクトリにコピーし，その後課題を終えたときに削除している作業が当てはまった．しかし，削除によって作業内容は他の学生と変わらないのにグラフの形状が結果大きく異なってしまった．直感的な学生の行動確認のためにはあまり適切な変動とはいえない．

6.3.5. コンパイルが止まっている学生

コンパイルが止まっていた 37 番の学生のグラフを図 4 に示す．授業中盤でコンパイルが止まっている．コマンドヒストリを確認することで，それ以外の作業もしていないであろうことがわかったので，寝ているなどの，集中が途切れていた状態であると予測できる．授業中に 37 番を確認できなかった理由として，他の学生の質問の対応に追われ，終盤グラフをあまり確認できなかったことがあげられる．このことからグラフの閲覧にタブレット端末の対応等を考慮すると，最終的にはシステムのグラフ閲覧作業は web サービスとする形が適しているといえる．また，試行を重ね，同じようなパターンを集めることでそれまでの行動に途中で集中を欠いてしまう傾向があれば，事前に教員が注意して行動できるため，授業への貢献が期待できる．

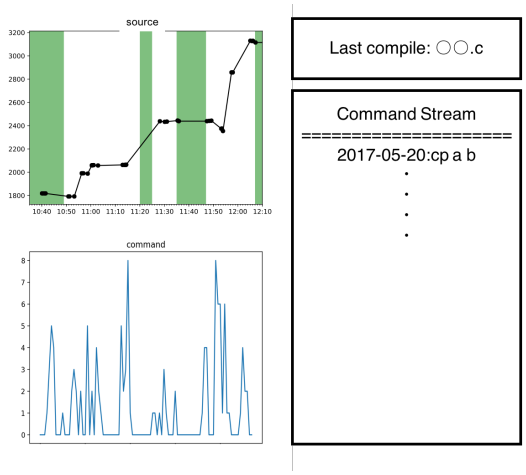

図 5 提案する新システムイメージ図

7. システムの改善点

主にソースコード量の急激な増加と高頻度のコンパイルに注目することで，学生がどのような状態にいるかはある程度予想可能であることはわかった．しかし講義と実習を交互に行うプログラミングの授業の性質上，終わっていない課題の回答や自主学習などがあり，急激な増加の回数によって今何をしているかを判断することなどは難しく情報が足りていないと言える．コンパイルの情報等から最後に作業していたものが何かを確認できるようにする必要が求められる．また，授業中はグラフの描画がされないとき，何らかの作業中なのか，思考中であるのか，集中を欠いているのかが即座にはわかりづらい．コマンドの発行についても可視化することで，コーディング中または集中を欠いている状態とそれ以外を一目で区別できるようになることが期待される．そして，6.3.4 項のとおり，不要ファイルの削除による可視化性の低下への対応も必要である．更に認識のしやすさを高めるためには，教員が説明中の時間を取得し，図 3 のように可視化情報として加えることで，同じ間隔でも容易に違う意味として捉えることができる．簡易的なイメージ図を図 5 に示す．

8. まとめ

本研究では，学校の授業という少数かつ変動が激しく，自動化の難しい要支援者の発見を目的とした授業の補助手段として学生のソースコードサイズがどのように変動するかを可視化することを検討した．

試行回数は少ないものの，ソースコードサイズ変化の可視化は声をかけるべき学生の判断にある程度実用性がみられた．しかし，現状では情報量が不足している．専用のコーディング環境を学生に強制しないことを 1 つの目的としているため，得られるログデータの詳細性には限界がある．しかし，複数のログデータを組み合わせ，補助的な自動アラート機能などを付け加えることで，教員の補助システムとしてより有用性が増すことが期待できる．

今後の展望として，必要な情報を可視化ツールに加えて表示するように改良するとともに，より簡単に情報にアクセスできるようにする．さらに，ログデータの傾向を統計・機械学習的アプローチでつかみ，自動的に支援が必要な学生を予測する機能を追加することで，教員が直接ログデータの可視化情報を見る必要性を下げることを検討することがあげられる．また現在は初学者への小規模なプログラミングによるアルゴリズムや言語の理解を深めるための課題の回答を対象として解析を行っているが，将来的には簡単なゲームプログラムなど中規模以上の場合にも利用価値があるのか，多言語ではどのようにログデータを取得し，可視化することが効果的かなどの検討が必要である．

謝辞

本研究は JSPS 科研費 16K01095 の助成を受けたものです．

参考文献

[1] 高田 義弘, 鳥居 宏次, “プログラマのデバッグ能力をキーストロークから測定する方法”, 電子情報通信学会論文誌. D-I, 情報・システム, I-コンピュータ, Vol.77-D-1(9), pp.646–655 (1994).

[2] 朽木 拓, 山田 敬三, 佐々木 敦, “プログラミングスキルレベル評価手法の研究”, 情報処理学会全国大会講演論文集, Vol.72, pp.521–522 (2010).

[3] “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, Vol.54 No.1, pp.330–339 (2013).

[4] 石嶋 慧, 平川 豊, 大関 和夫, “ウェブブラウザを用いたプログラミング学習支援環境” , 報処理学会全国大会講演論文集, Vol.78,pp.349–350 (2016).

[5] 清野 真理子, 橋本 玄基, 大枝 真一, “Random Forest を用いたコマンド履歴からのプログラミングスキル推定”, 情報処理学会全国大会講演論文集, Vol.78, pp.895–896 (2016).

[6] 橋本 玄基, 清野 真理子, 大枝 真一, “プログラマのスキル評価のためのログデータ解析”, 情報処理学会全国大会講演論文集, Vol.78, pp.899–900 (2016).

[7] 橋本 玄基, 大枝 真一, “プログラミング初学者教育における要支援者予測のためのログデータクラスタリング解析”, 情報処理学会全国大会論文集, Vol.79 (2017).

[8] Shinichi Oeda, Genki Hashimoto, “Log-Data Clustering Analysis for Dropout Prediction in Beginner Programming Classes”, 21st International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES2017), (to appear).