K-014

学習履歴を活用したデバッグ練習問題抽出システムの開発
Development of Debugging Exercise Extraction System using Learning History

梅澤 克之† 中澤 真‡ 後藤 正幸†† 平澤 茂一††
Katsuyuki Umezawa Makoto Nakazawa Masayuki Goto Shigeichi Hirasawa

1. はじめに

近年，学習モデルは多様化し，e ラーニングから，教場授業と e ラーニングを併用するブレンディッドラーニングまで多岐にわたっている．このような教育環境においては，LMS（学習管理システム）が利用される．LMS は学習コンテンツの配信，試験の実施と採点など様々な機能を持ち，また LMS を経由して学習者の学習活動に関する多くの履歴（操作ログなど）・記録（レポート・評価結果など）の収集が可能である．

我々は，プログラミング言語学習の学習環境である編集履歴可視化システムを提案してきた[1]．このシステムは学習環境の準備を容易にするとともに学習者の状況把握することができる．また学習ログを蓄積することで学習者がプログラムのどこをどのように修正したかを確認できる．また我々は，自習時の学習ログに基づき対面授業時にグループ分けをして授業を行うグループ分け反転授業を提案してきた[2][3]．この反転授業を実授業に適用するにあたり，前述の編集履歴可視化システムを活用した[4][5]．その結果，約 90 名の学生が 16 週間の授業を受けた際の膨大な学習ログが蓄積された．

プログラミングの授業を実際に行ったところ，エラー表示がされているにもかかわらず，そのエラー表示を読もうともせずに，教員にヘルプを求める学生がとても多い事がわかった．

編集履歴可視化システムでは，プログラムが完成するまでに修正されていく過程のソースコードをすべてログとして蓄積している。これらの情報を元にあえて間違いを含むソースコードを学習者に与えて，そこに含まれている間違いを修正するデバッグ練習用の問題を自動生成することが可能であると考え，開発を行った．

本報告では，編集履歴可視化システムが蓄積した学習データをもとに，デバッグ練習用の問題を自動生成するデバッグ練習問題抽出システムについて記述する．

2 章では，関連技術として，編集履歴可視化システムの概要および反転授業への活用について述べる．3 章で開発したデバッグ練習問題抽出システムについて，画面構成，抽出アルゴリズム，および抽出後の可視化について記述する．4 章でまとめと今後の課題を示す．

2. 関連技術

2.1 学習履歴の蓄積システム

学習者がプログラムを完成させて行く過程や，その過程でどのような課題に直面し，どのように解決したかを知ることは難しく，それらを理解するためにはバージョン単位の履歴では不十分であり、どのように変更されてきたのかユーザの操作単位で詳細に記録を取ることの有効性が指摘されている[6][7]．文献[6]では，蓄積した学習履歴の活用として，操作履歴を表示する機能，表示機能に対して表示条件を設定してフィルタリングする機能，過去の任意の状態のソースコードを表示・復元する機能を提案している．文献[7]では，詳細な学習履歴をリアルタイムに蓄積し分析することで，該当の学習者への良い点，悪い点のフィードバックを可能にし，教師側は学習者の躓きの箇所がわかる機能を有しているが，蓄積した学習者全員の履歴を自動で分析したり，分析結果をもとに品質を向上させるまでには至っていない．

2.2 編集履歴可視化システム

我々も 2.1 節に示したシステムと同様のコンセプトに基づいて，プログラミング言語学習の学習環境である編集履

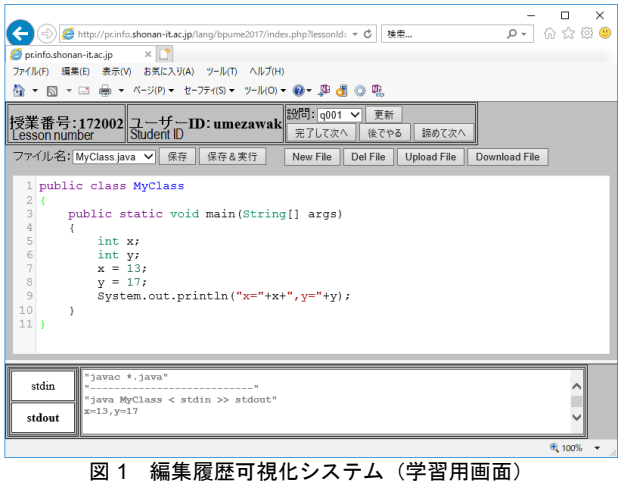

図 1　編集履歴可視化システム（学習用画面）

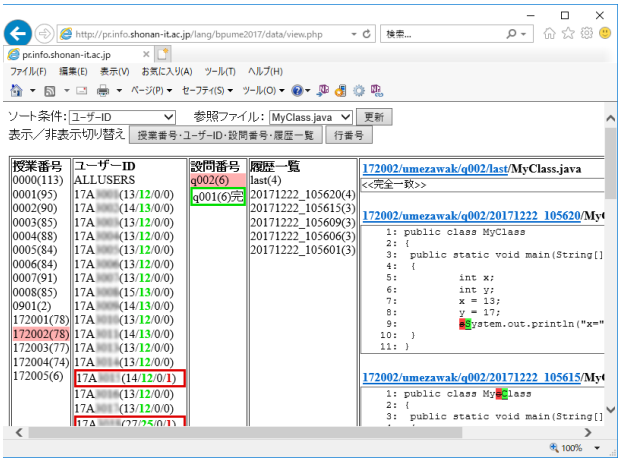

図 2　編集履歴可視化システム（教師用画面）

† 湘南工科大学　Shonan Institute of Technology
‡ 会津大学短期大学部　Junior College of Aizu
†† 早稲田大学　Waseda University

歴可視化システムを提案してきた[1].

編集履歴可視化システムは，プログラミング言語学習（英語学習にも適用可能）の初学者に特化した学習環境である．このシステムは，学習環境の準備の容易化，学習者の状況の把握などの課題を解決するためのシステムである．「学習環境の準備の容易化」に関しては，開発環境のインストールなどの前準備は不要で，ブラウザから利用可能となっている．そのため学習者は，PC だけでなくスマートフォンやタブレットからの学習も可能である．また「学習者の状況把握」に関しては教師用の画面でソースコードや実行結果の 1 つ前の状態に対する差分を表示可能である．この差分を見れば学習者がどこをどう修正したかを確認できるようになっている．また，それぞれの学習者が何問目の問題を解いている最中なのかを一目で確認できるようになっている．開発可能なプログラミング言語は、C/C++[8], Java, HTML/JavaScrip, Scratch[9]などである。また英語のライティング問題にも対応している[10]．Java 言語版の編集履歴可視化システムの学習用画面を図 1 に，教師用の確認画面を図 2 に示す．

2.3 反転授業で活用

我々は，反転授業の自習時の e-ラーニングの学習ログを取得し，自習時の理解度が高い学生のグループ，自習に時間をかけなかったために理解度が低い学生のグループ，自習に時間をかけたが理解度が低い学生のグループに分けて教場での対面授業を行う「グループ分け反転授業」を提案してきた[2][3]．このグループ分け反転授業の方式を 2017 年度後期における 16 週間の実授業「基礎プログラミング実習」に適用し，自習後の理解度と対面授業後の理解度の観点でその有効性を示してきた[4][5]．この反転授業の実授業への適用の中で，2.2 節で示した Java 言語用編集履歴可視化システムを活用し，前半の 8 週間の 98 名分，および後半の 8 週間の 85 名分の膨大な学習ログを蓄積した．

3. デバッグ練習問題抽出システム

3.1 システムの全体構成

今回開発したデバッグ練習問題抽出システムの全体構成を図 3 に示す．図 3 に示す通り，学習履歴を蓄積するまでは既存システムである編集履歴可視化システムを活用する．デバッグ練習問題抽出ツールは，編集履歴可視化ツールが蓄積した学習履歴を参照し，正解のソースコードと正解に至るまでの誤りを含むソースコードを比較し，誤りを含むソースコードを抽出する．抽出された誤りを含むソースコードは，編集履歴可視化システムの改造版で参照及び配布される．

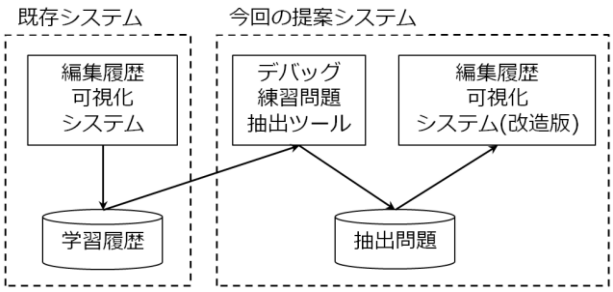

図 4　デバッグ練習問題抽出システムの全体構成

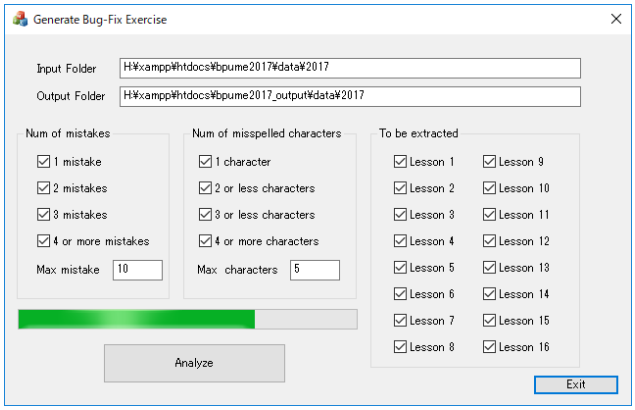

図 3　デバッグ練習問題抽出ツール

3.2 デバッグ練習問題抽出ツール

デバッグ練習問題抽出ツールは，編集履歴可視化ツールが蓄積した学習履歴，具体的には，各問題ごとの完成フォルダ（last）に存在する MyClass.java と，作成途中の MyClass.java を比較し，修正箇所数および各修正箇所の修正文字数を抽出する．抽出した「修正箇所数」および「最大修正文字数」をもとにフォルダを再構成する．なお，ソースコードの差分には，エラーを修正したための差分と、エラーを伴わない差分があるため，エラーを修正した場合のソースコードのみ抽出する事としている．開発したデバッグ練習問題抽出ツールを図 4 に示す．

図 4 に示すように，何個までの修正箇所のものを抽出するか，何文字までの修正文字数のものを抽出するか，どの授業の問題を抽出するか等を指定出来るようにしている．

3.3 抽出アルゴリズム

本節では抽出アルゴリズムを示す．

1) 図 5 のフォルダ構成の全履歴について下記を繰り返す．
2) last.info に“end”と記載されているかを確認する．
3) last フォルダ以外のフォルダについて，stdout ファイルに“エラー”が記載されているかどうかを確認する．
4) last フォルダ以外のエラーが記載されているフォルダの MyClass.java と last フォルダの MyClass.java の差分をとる[1]．
5) その際に，差分の箇所の数と 1 つの差分に最大何文字含まれているかをカウントする．
6) 差分の箇所数と 1 つの差分に最大何文字含まれているかによって図 6 のフォルダ構成に従って MyClass.java ファイルをコピーする．

3.4 抽出にかかる実行時間の評価

今回の 16 週間分のプログラム編集履歴データの総数は 31,562 ファイルであり，その中でエラーを伴うものは 18,680 ファイルであった．表 1 に示すコンピュータで図 4 に示す抽出オプションでデバッグ練習問題抽出ツールを実

[1] Java 版編集履歴可視化システムでは，main 関数を含むクラス名は MyClass でありそのファイル名は MyClass.java と規定されている．

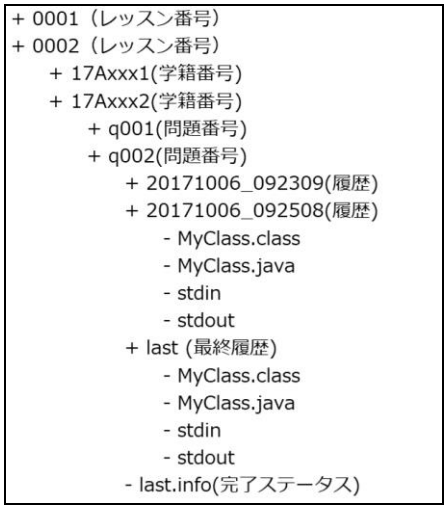

図 5　編集履歴可視化システムのフォルダ構成

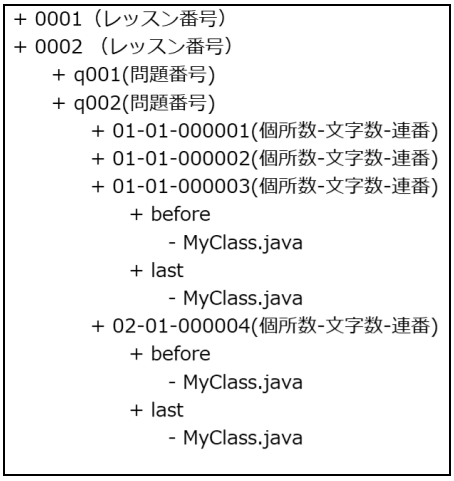

図 6　デバッグ練習問題抽出ツールが出力するフォルダ構成

行したところ，抽出が完了するまでの実行時間は 488 秒となった．約 100 名の半期の授業の全学習履歴を一度だけ分析すれば良いということを考慮すると十分実用的な実行時間ということができる．

表 1 実測を行ったコンピュータのスペック

CPU	Intel(R) Core(TM) i5-2400 CPU @ 3.10GHz
メモリ	16GB
OS	Windows 10 Pro (64 bit)

3.5 抽出データの可視化と配布

図 7 に示すように，抽出したデータは編集履歴可視化システム（改造版）で可視化できる．図 7 は，授業番号 5 の設問 4 の 2 か所の間違い，かつ最大 2 文字の間違いを含むソースコードが示されている．編集履歴可視化システムはソースコードの配布機能も付いているので，このソースコードを学生に配布して間違いを修正させる問題を出題することができる．

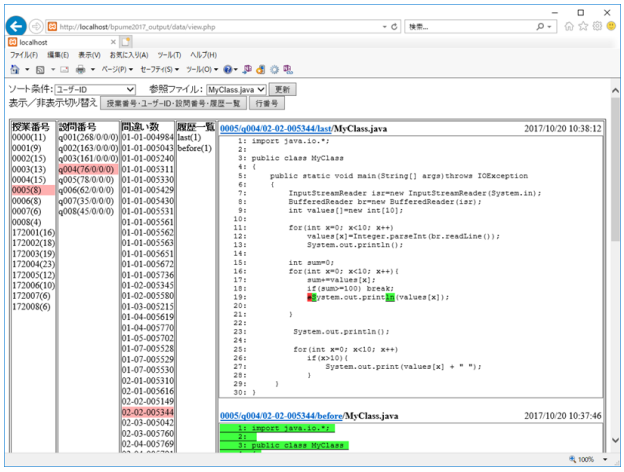

図 7　抽出したデータの可視化

4. 今後の拡張計画

3.3 節で示したアルゴリズムでは，間違い易さまで考慮されていない．同種のエラーを判定するステップを追加し，エラーの頻度を算出することで，間違い易い問題を抽出できるのでより効果的な授業を行うことが可能と考えられる．同種のエラーの判定方法としては，下記のようなものが考えられる．

- エラー種別と差分の箇所数と文字数と修正箇所の場所まで完全一致
- エラー種別と差分の箇所数と文字数が一致（場所違いは問わない）
- エラー種別と差分の箇所数が一致（文字数と場所の違いは問わない）
- エラー種別と差分の文字数が一致（箇所数と場所は問わない）
- エラー種別が一致（文字数と箇所数と場所は問わない）

5. まとめと今後の課題

16 週間の実授業において，学習ログを蓄積することで学習者がプログラムのどこをどのように修正したかを確認できる編集履歴可視化システムが蓄積した学習ログをもとに，デバッグ練習用の問題を自動生成するデバッグ練習問題抽出システムを開発した．

今後は，4 章で述べた間違い易さまで考慮した抽出方法を確立するとともに，このシステムを使って作成したデバッグ練習用の問題を使ってプログラミング学習を行い，デバッグ練習を行わない場合と比較して教師への依存度が低減することを実証していきたい．

謝辞

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 (B)19H01721，(C)17K01101，(C)16K00491,早稲田理工研特別勘定 1010000175806 NTT 包括協定共同研究，および，経営情報学会「ICT と教育」研究部会の助成による．本研究成果の一部は早稲田大学理工総研プロジェクト研究「次世代 e-learning に関する研究」の一環として行われたものである．

参考文献

[1] 荒本道隆，小林学，中澤真，中野美知子，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～システム構成と実装,” 情報処理学会第 78 回全国大会, 横浜市,pp.4-527–528, Mar. 2016.

[2] Katsuyuki Umezawa, Manabu Kobayashi, Takashi Ishida, Makoto Nakazawa, and Shigeichi Hirasawa,“Experiment and Evaluation of Effective Grouped Flipped Classroom,” Proceeding of the 5th International Conference on Applied Computing & Information Technology (ACIT 2017), pp.71-76, July 2017.

[3] Katsuyuki Umezawa, Manabu Kobayashi, Takashi Ishida, Makoto Nakazawa, and Shigeichi Hirasawa, “Use of Student Grouping to Make Flipped Classroom More Effective,” Proceeding of the 16th Hawaii International Conference on Education, p.p. 1249-1250, Jan. 2018.

[4] 梅澤克之，小林学，石田崇，中澤真，平澤茂一，“グループ分け反転授業の実授業への適用，” 電子情報通信学会教育工学研究会(ET) 予稿集, pp.199-204, Feb. 2018.

[5] Katsuyuki Umezawa, Takashi Ishida, Makoto Nakazawa and Shigeichi Hirasawa, “Application and Evaluation of Grouped Flipped Classroom Method to Real Classes,” Proceeding of the International Conference on Engineering, Technology, and Applied Science (ICETA2018), p.99, June 2018.

[6] 大森隆行，丸山勝久，“開発者による編集操作に基づくソースコード変更抽出,” 情報処理学会論文誌, Vol.49, No.7, pp.2349-2359, 2008.

[7] 森一樹, 田中昂文, 橋浦弘明, 櫨山淳雄, 古宮誠一，“プログラミング演習支援のための細粒度履歴収集環境の開発,” 情報処理学会研究報告（SE）, 179(16), pp.1-6, 2013.

[8] 小林学，後藤正幸，荒本道隆，平澤茂一，“プログラミング編集履歴可視化システムとその実践,” 日本経営工学会秋季大会予稿集，pp.8-9, Nov. 2015.

[9] 中澤真，後藤正幸，荒本道隆，平澤茂一，“ビジュアルプログラミング言語「Scratch」のための学習履歴分析環境とその可能性－初等教育からのプログラミング教育に向けてー,” 日本経営工学会秋季大会予稿集，pp.10-11, Nov. 2015.

[10] 中野美知子，荒本道隆，吉田論史，“プログラミング言語の学習ログ収集ソフトウェアを活用した文法矯正練習の試み，” 日本経営工学会秋季大会予稿集，pp.12-13, Nov. 2015.