1F1-OS-26a-2

プログラミング演習における演習履歴自動収集環境とそれを利用した授業改善の枠組み

An automatical collection environment of students' exercise history on programming class and a framework of programming class improvement using the environment

小暮 悟 *1 Satoru Kogure　杉山 匠 *2 Takumi Sugiyama　野口 靖浩 *1 Yasuhiro Noguchi　小西 達裕 *1 Tatsuhiro Konishi　伊東 幸宏 *3 Yukihiro Itoh

*1静岡大学 情報学部
Faculty of Informatics, Shizuoka University

*2静岡大学 大学院 総合科学技術研究科 情報学専攻
Department of Informatics, Graduate School of Integrated Science and Technology, Shizuoka University

*3静岡大学
Shizuoka University

In typically, educational institutions such as universities conduct class questionnaires to obtain evaluations for classes. A class questionnaires is used to obtain all lectures' (15 times in typically) evaluations, and the questionnaires is not used to each lecture's evaluation. We had developed a support system for programming class to observe each student's behavior on each lecture. The system support a students' refection of own programming behavior, teacher's desk patrol/guidance on a class and teacher's class improvement after class. In this study, we use the system to evaluate a difficulty of each exercise of each lecture and discuss the class improvement to utilize our systems.

1. はじめに

古くから大学等の教育機関において，学生の授業に対する満足度などの主観評価を得るために，授業アンケートが実施されている．近年，授業アンケートの収集はマークシートへの記入後の OCR による電子化や，LMS 等の Web サイトを経由した電子化が進んでいる．授業アンケートは基本的には講義全体の満足度や自身の授業内容にたいする理解度の自己評価などの主観的評価の側面が多くを占める．また，授業全体の評価であり，個々の講義への評価を得ることはできない．個々の講義への評価に関しては，それぞれの講義の後に別途アンケートや小テストを実施したり，コメントペーパー等を利用することなどが考えられるが準備や集計などにコストがかかる．

一方で授業中の個々の学生の理解状況をまとめ上げクラス全体として汎化し，全体への説明が必要か判断する情報を自動収集する研究も幾つか行われている [内藤 2008]．我々も，プログラミング講義・演習を対象とした講義・演習支援システムを構築している [Kogure 2015]．このシステムは，プログラミング演習中に，学生がいつの講義 L_i のどの課題 E_{ij} のどのステップ S_{ijk} を今実施中かを自動判定できる．このシステム利用時には，教師は 予めある講義 L_i における全課題・全ステップの正解プログラムに対応する PAD(Problem Analysis Diagram) 形式の正解アルゴリズムを準備しておく．システムは，学生が作成したプログラムを PAD に変換して全正解アルゴリズムとマッチングし，最も一致率が高いもの (つまり完成している) の次のステップ，もしくは次の課題の最初のステップに取り組んでいると判定する．これにより，教師は想定していた演習進度と実際の学生の演習の進み度合いを比較することでクラス全体へのフォローが必要か判断できる．正解アルゴリズム (PAD) と学生の作成したプログラムを変換した PAD のマッチングには，我々がこれまでに開発してきた，プログラム評価器を利用している [鈴木 2007].

また，授業中の教員・TA のサポートを目的とした授業支援システムも開発している [Noguchi 2016]．こちらも前述の研究と同様，プログラム評価器を利用する [鈴木 2007]．教員・TA はシステムを利用し，個別に指導が必要な学生がいるか判断し，いると判定された場合は予め入力しておいた座席表を見て個別指導に向かう．個別指導中，もしくは個別指導が終わったあとに教員・TA は個別指導の内容をシステムに入力する．この情報はあとで別の教員・TA が再利用可能である．頻発する質問等では他の教員・TA も含めた過去の個別指導の際に利用した資料を再利用したり，指導対象の学生の指導履歴をあらかじめ見ることで，均一な (もしくはより洗練された) 個別指導を行ったりこれまでの指導履歴を踏まえて個別指導を行える．

これらの研究の考え方を発展的に改良し，学生の演習中の躓きの検出をベースとした「1. 演習後に学生が自身の演習を振り返る」「2. 演習中に教員がクラス全体の状況を把握する」「3. 講義・演習終了後に教員が前回の課題の進捗状況もしくは昨年度の課題の進捗状況を総合的に判断し，講義改善に利用する」ための枠組みを提案した [杉山 2017]．実際に学生の演習中のつまづきの検出のためのシステムを構築し，情報系大学 1 年生を対象とした 100 名規模の講義で実際に導入した．実施 6 週間 (6 回の講義分) において，学生の躓きの検出の精度 0.549, 再現率 0.824 を得た [杉山 2017]．本報告においては，本システムの授業改善への利用方法を検討したので報告する．

2. 迷い検出に基づく学習教育支援システム

迷い検出に基づく学習教育支援システム [杉山 2017] の概略を述べる．

本システムは，プログラムの編集履歴，コンパイル時のソースコード，コンパイルエラー，実行時エラー (以降コーディング情報と記す) を自動収集し，経験則に基づく判定基準に従って学習者の迷い (以降 impasse と表記する) を impasse 検出器により検出する．検出された impasse を「学生が演習終了後に使用する」振り返り支援システム,「教師が演習中に使用する」机間巡視支援システム,「教師が演習終了後に使用する」講義設計支援システムの 3 つで利用することが想定される．以

連絡先: 小暮 悟, 静岡大学情報学部
〒 432-8011 静岡県浜松市中区城北 3-5-1
E-mail: kogure@inf.shizuoka.ac.jp

下「impasse 検出器」と「3 つのシステム」について述べる．

2.1 impasse の検出

impasse 検出にあたり，まず「どのような impasse を検出するのか」「それぞれの impasse はどのような要因で検出できるか」を検討した．これについては，プログラミング経験のある情報系大学 4 年生 5 名の被験者に，延べ人数 15 人，合計 30 時間のプログラミングを行ってもらった．impasse を観察する目的のため，プログラミング課題として，比較的難易度の高いもの (国際大学対抗プログラミングコンテスト ACM-ICPC の過去問題) を与えた．impasse の観察のため，プログラミング中の状況を操作画面の録画および直接の観察により把握した．また，impasse を判断する要因の収集のため，コーディング情報も収集した．

まず，impasse は以下の 4 通りに分類できると判断した．また，コーディング情報も見ることで，それぞれの impasse の原因となる情報 (以降，迷い兆候と呼ぶ) も検討した．コーディング情報の観測器が迷い兆候を監視し，以下の条件が満たされたら，検出された迷い兆候と，それに対応する impasse を出力するよう実装した．

- impasse(1) **実装方針がたたない**
  迷い兆候 (1-1) 15 分以上ソースコードが未変更
- impasse(2) **コンパイルエラーが解決できない**
  迷い兆候 (2-1) 前回のコンパイルからエラー減少なし
  迷い兆候 (2-2) 解決済みコンパイルエラーが再出
- impasse(3) **実装時エラーが解決できない**
  迷い兆候 (3-1) 同一の実行時エラーが 7 回以上出現
- impasse(4) **論理エラーが解決できない**
  迷い兆候 (4-1) 同一箇所の編集
  迷い兆候 (4-2) 変数の標準出力関数の追加，削除
  迷い兆候 (4-3) 3 分以内に 2 回以上コンパイル
  迷い兆候 (4-4) 過去の状態にソースコードが巻き戻り
  迷い兆候 (4-5) ソースコードが 15%以上変更
  迷い兆候 (4-6) 同一の変数を含む箇所を 3 回以上連続で編集

2.2 検出した impasse の活用

2. 章で示した 3 つのシステムについて個別に説明する．

2.2.1 学生の振り返りへの活用

学生は，自身の理解状況を把握するために振り返り用のシステムを利用する．学生は自身が演習中にどういった impasse に陥り，その前後にどのような編集を行っていたかを確認して振り返りを行う．演習中に学生が陥っていた impasse 全てを列記すると，学生はどの impasse から自身の演習を振り返えればいいか混乱すると考える．そこで，生じた impasse の中でも特に重要度の高いものやより長い時間生じていたものに着目させる．これには，演習中の impasse の出現度合いをヒートマップ形式で表示することにより，注目すべき時間帯を学習者が判断できるようにインターフェースを設計した．詳しくは文献 [杉山 2017] を参照のこと．

2.2.2 教師の机間巡視への活用

机間巡視用のインターフェースは未実装であり，構想をまとめる．机間巡視では，教師が impasse に陥っている学生を見つけ， その学生のコーディング履歴や impasse 履歴から指導を行う手助けできることが望ましい． そのためには，現在どの学生が impasse に陥っている疑いがあるのか，impasse に陥っている学生がどういった編集を行ってきたかといった情報が必要となる．現在 impasse に陥っている可能性のある学生を抽出して表示し，その学生の過去の編集履歴を見ることができるインターフェースを用いて支援することを考えている．

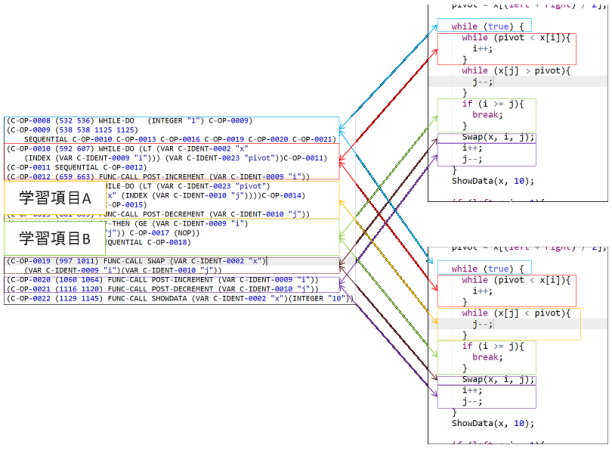

図 1: 学習項目と impasse 発生箇所の対応付け例

2.2.3 教師の講義設計への活用

本稿の主題であるため詳細に説明する．

講義設計を行う際，教師は学生に出す課題難易度の調整をおこなう．課題解決に苦労した学生がどういった impasse に陥ったかの傾向を知ることができれば次回のレポート課題の難易度を現在の学習者のレベルに適した形で調節できる．また，すでに教えている内容でフォローが必要な事項を知ることができる．課題達成が困難だった学生はより長時間 impasse に陥っていたと考えられるため，収集したコーディング情報を基に課題に掛けた時間や検出した impasse の時間を基にソートすることで課題達成が困難だった学生を見つけることができるはずである．ソートによってより長い時間 impasse に陥った学生を抽出し，その学生が impasse に陥った時間帯の編集履歴を確認できるインターフェースを作成することにより支援を行う．また，多くの学生がどういった学習項目について impasse を生じているのかを知ることで，教師は次回以降の講義における説明内容を考えられる．しかし，この情報を教師に示すためには学生のソースコードの impasse が生じた箇所と，学習項目の対応付けが必要である．そこで，先行研究であるプログラム半自動評価システム [鈴木 2007] を用いて対応付けを行う．

図 1 に対応付けの仕組みを示す．教師は予め，学習項目タグを正解となる標準アルゴリズムに埋め込んでおく ;．また，impasse 検出器によって， impasse 発生時のソースコードがを特定可能である．図 1 の α が impasse 発生時に学生のプログラム， β が impasse 解消時の学生プログラムである．これにより，学生がプログラムのどの箇所で詰まっていたかがわかる．そして，予め埋め込まれた学習項目タグを見ることで，どの学習項目で詰まっていたのかを判断できる．

3. 演習内容自動収集システムを利用した授業改善の枠組み

演習内容自動収集システムを利用した授業改善の枠組みとしては，「(a) コーディング情報の自動収集」「(b) 各学習項目に

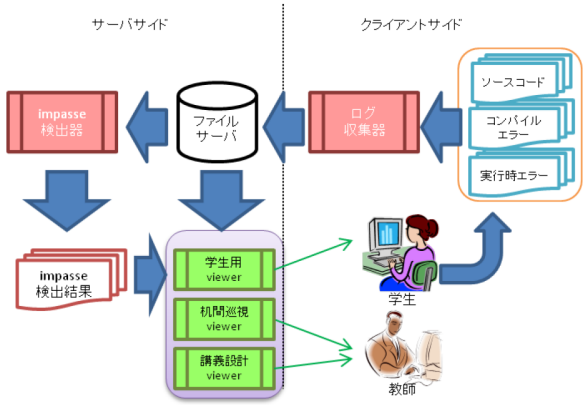

図 2: システム構成図

おける学生の躓き傾向把握」「(c) 演習における各課題の難易度検出」「(d) 授業改善案の作成」を考える．(a) 及び (b) についてはすでに述べた先行システム [杉山 2017] を利用する．以降，(a), (b) についての簡単な説明をした後，(c), (d) について説明する．

3.1 コーディング情報の自動収集と各学習項目における学生の躓き検出

システム構成図を図 2 に示す [杉山 2017]．まず「ログ収集器」は，学生の演習環境でサーバとして起動される．本システムを実際に使ってもらった講義では，プログラミング演習はコンソールで行っていたためそちらで対応できるような実装を行った．対象言語は Java であり，コンパイラ javac と 実行環境 java を，こちらで用意したスクリプトでラッパーさせた．これにより，学生は通常の演習を進めるだけで，コンパイル時のソースコードやコンパイルエラー，実行時エラーを収集できる．また，予め編集予定のクラス名を入力してもらうことで，クラス名.java のファイルを監視し，変更があったら収集するよう実装を行った．これによりソースコードの編集履歴を収集できる．「impasse 検出器」は収集したコーディング情報と設定した迷い兆候による検出ルールを入力し，impasse の検出を行う．

3.2 各学習項目における学生の躓き傾向把握

本機能は，2.2.3 項で述べたプログラム評価器による正解アルゴリズムと学習者プログラムの対応関係を必要とするため，予め全学生プログラム及び正解プログラムを PAD 化し，学生プログラムの正誤判定を行っておく必要がある．また，教師は，表 1 のようなプログラムのステートメント行番号に対応した学習項目をあらかじめ設定しておく必要がある．この情報を csv で保存し，図 3 上でその csv ファイルを読み込ませることで，多くの学生が詰まったプログラムの行が強調表示され，また，その行に対応する学習項目も多くの学生が躓いた学習項目の色が濃く表示され，クラス全体でどの学習項目が躓きやすいかの傾向を知ることができる．

3.3 演習における各課題の難易度検出

この節に関してはまだ検討段階でありシステムの実装には至っていないが検討中の内容について記す．

本研究で利用しているコーディング情報は，演習中の学生の躓きを検出する目的に利用している．しかし，それ以外にも様々な情報が利用できると考えられる．また，先行研究 [Kogure 2015]

表 1: 正解プログラムの行番号とそれに対応した学習項目の登録例

行番号	学習項目
10	標準入力
15	while ループ
15	比較演算子
:	:

のシステムの実装を利用することで，各学生がどの課題まで取り組んだかを把握することが可能である．これらの情報を融合的に利用することで，課題ごとのアンケートを取ることなく学生の各課題に感じた難易度を推定することができると考えている．

予備的な調査として，3 個の課題について，「ソースコードの編集時間」「コンパイル回数」「実行回数」の 3 変数を従属変数とし，該当する演習における学習者自身の難易度評価の課題ごとのアンケート結果の「評点 (4. 楽勝，3. なんとかできた, 2. 時間をかければできそう, 1. 絶対無理)」を目的変数として重回帰分析を行った．重相関はそれぞれ 0.39, 0.27, 0.27 と，相関がないという結果を得た．例えば，ソースコードの編集時間において考えると，時間が短かった場合は「簡単ですぐに終わった」場合と「少しやってみたが難しくて諦めた」場合を分別できないし，コンパイル回数が多かった場合においても「コンパイルエラーが頻発して頑張ってデバッグをした」場合と「作成したメソッドをテストするために様々な初期データを与えてコンパイル実行した」場合が分別できないと考える．

こちらに関しては，今後，(b) の躓きの分布を取得する際の，各学生の impasse の種類や回数を従属変数において，課題の難易度の推定ができないか検討を行っていく．

3.4 授業改善案の作成

(b) によって，学習者の躓きの傾向が，(c) によって各課題への学習者の印象 (難易度) が推定できれば，次週，どのようなフォローが必要で，課題の難易度の調整が必要かを予め予測することが可能となる．現時点ではまだ授業改善案は教師が様々な情報を見て自分で作成する状況となっているが，今後は，授業改善を提案するシステムに関する研究も進めていきたい．

4. まとめ

「1. 演習後に学生が自身の演習を振り返る」「2. 演習中に教員がクラス全体の状況を把握する」「3. 講義・演習終了後に教員が前回の課題の進捗状況もしくは昨年度の課題の進捗状況を総合的に判断し，講義改善に利用する」ためのシステムの枠組みと，1 のシステムの実装が先行研究にて実施された．本稿では，3 のシステムを実装し，そのシステムが授業改善にどのように利用できるか述べた．また，演習内容に対して学生が感じる難易度の推定方法の検討と，impasse の検出結果や課題に対する難易度検出結果を授業改善に活かす方向性について検討した．

今後は，impasse の検出結果からの課題に対する難易度の検出方法の検討と，授業改善を提案するシステムを模索していく．

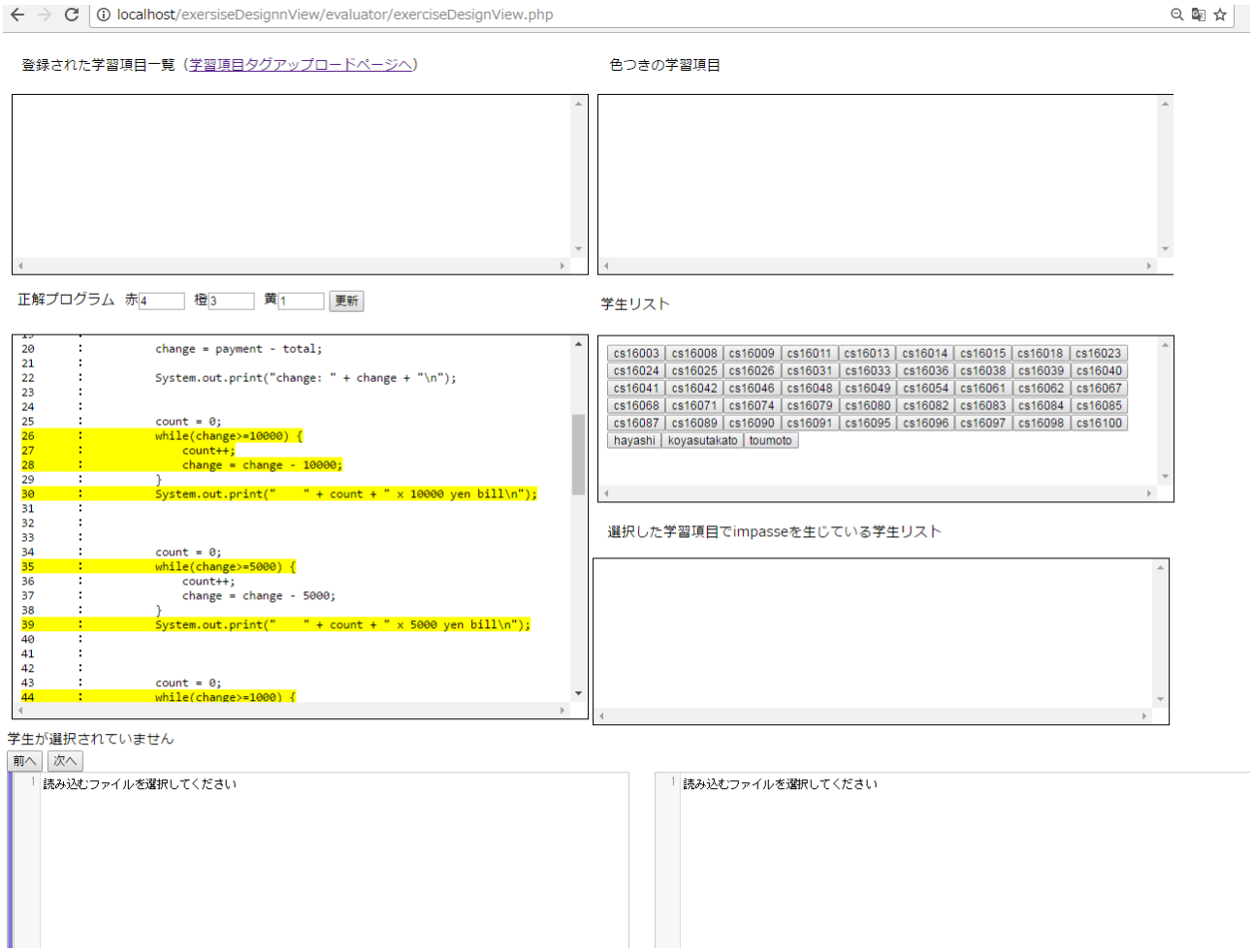

図 3: 各学習項目における学生の躓き傾向を把握するシステム

謝辞

本研究は JSPS 科研費 24300282 の 助成を受けたものである.

参考文献

[内藤 2008] 内藤 広志, 斎藤 隆: プログラミング演習のための進捗モニタリングシステム, 情報処理学会, コンピュータと教育研究会報告, 2008-CE-93, pp.33-40 (2008).

[Kogure 2015] Kogure, S., Nakamura, R., Makino, K., Yamashita, K., Konishi,T., Itoh, Y.: Monitoring System for the Effective Instruction based on the Semi-automatic Evaluation of Prorams during Programming Classroom Lectures, Research and Practice in Technology Enhanced Learning (RPTEL), Vol.10, No.18, pp.1-12 (2015).

[鈴木 2007] 鈴木 浩之, 小西 達裕, 伊東 幸宏: 抽象的データ構造を含むアルゴリズム表現に基づくプログラム評価支援システムの構築, 教育システム情報学会誌, Vol.24, No.3 ,pp.167-186 (2007).

[Noguchi 2016] Noguchi, Y., Osawa, R., Yamashita, K., Kogure, S., Konishi, T., Itoh, Y.: Networked Tutoring Support System for a Programming Class based on Reusable Tutoring Content and Semi-Automatic Program Assessment, Proceedings of International Conference on Computers in Education (ICCE2016), pp.252-257 (2016).

[杉山 2017] 杉山 匠, 小暮 悟, 山下 浩一, 野口 靖浩, 小西 達裕, 伊東 幸宏: プログラミング演習における迷い検出に基づく学習教育支援システムの構築, 電子情報通信学会技術報告, Vol.116, No.438, ET2016-93, pp.81-86 (2017).