Vol.2020-CE-154 No.7
2020/3/14

レスポンスアナライザーの開発・応用に関する研究

近藤 郁哉[1]　海老原 宏樹[1]　高野 辰之[2]　小濱 隆司[3]　宮川 治[3]

概要：プログラミング教育において，教授者が学習者にプログラム作成の演習問題を出題することがある．プログラミングの演習では教授者が学習者の画面を確認し，学習状況の確認を行うことがある．しかし，学習者の画面から行き詰まっていることがわかりにくいため，指導が必要な学習者の発見が遅れ，指導に要する時間が長くなる．この問題を解決するために本研究では，演習問題に対する学習者の学習状況をレスポンスアナライザーを用いて取得し，リアルタイムで教授者に学習進度を提示するシステムを開発した．このシステムを授業に導入することにより，学習者全体の解答状況と学習者の個別の状況を教授者に提示することができ，授業速度の最適化が期待される．また，システムを実際の授業に導入して学習者が利用する実験をし，システムの評価と提出されたファイルの分析を行った．本稿では，システムの詳細と実験，提出されたファイルによる分析結果に関して報告する．

Study on development and application of response analyzer

1. はじめに

プログラミング教育において，教授者が学習者にプログラム作成の演習問題を出題することがある．プログラミングの演習では，遅れていていたり停滞しているにも関わらず，質問しない学習者がいる [1]．そのため，教授者は学習者の自発的な問合せに対応するだけでなく，学習状況に遅延や停滞状態にある学習者を把握し，状況に応じた指導を行うことが重要となる [2]．

プログラミング演習では机間巡視を行いながら学習者の画面を確認し，学習状況の確認を行うことがある．しかし，机間巡視では学習者の画面から行き詰まっていることがわかりにくいため，指導が必要な学習者の発見が遅れ，指導に要する時間が長くなる．また，教授者が学習者の画面から行き詰まっていることを発見できなかった場合，指導することができない．机間巡視では指導に必要な時間が長く，クラス全体の学習進度や指導を必要としている学習者の把握が困難であり，それに伴う授業進度の最適化も行うことができないという問題がある．

そこで本研究では，演習問題に対する学習者の学習進度をレスポンスアナライザーを用いて取得し，リアルタイムで教授者に学習進度を提示するシステムを開発する．学習者全体の解答状況と学習者の個別の状況を教員に提示することにより，授業速度の最適化を支援する．

2. 関連研究

プログラミング入門教育において，学習者もしくは教授者を支援するシステムはこれまで数多く研究されてきている．西田らは，初学者教育用プログラミング環境 PEN と呼ばれる学習環境を構築し，評価を行っている．このシステムでは，日本語をベースとした xDNCL と呼ばれる手順記述言語を用いて，プログラミングを分かりやすく学習者に教授しようと試みている [3]．斐品らは，アルゴリズム学習支援システムを提案している．このシステムでは，プログラミング言語によらずにアルゴリズムの理解を促すために，JPADet と呼ばれるツールを用い，抽象化されたアルゴリズムを図式的に学習者に解答させることができる [4]．これらの環境やシステムは，主に学習者の支援を目的としており，筆者が開発したシステムとはその目的が異なる．本研究で開発したシステムでは教授者と学習者の両者の支援を目的としている．プログラムの作成過程を「コンパイル」「インデント」「枠組みの導出」「実装」にわけ，どの過程

[1] 東京電機大学大学院 情報環境学研究科
Graduate School of Information Environment, Tokyo Denki University
[2] 関東学院大学 理工学部
College of Science and Engineering, Kanto Gakuin University
[3] 東京電機大学 情報環境学部
School of Information Environment, Tokyo Denki University

ⓒ 2020 Information Processing Society of Japan

でのエラーが多いのかを分析する．また，作成に長時間を要している学習者の原因分析を行う．これらの結果は教授者の講義内容の改善に大きく寄与するものと考えられる．

知見らは，学習者に内省を行わせることで，プログラミングの知識を定着させる学習環境を構築している．学習者のコンパイルエラー，実行エラー，あるいは学習者自身が入力する論理エラーなどの失敗情報を元に，内省を行う環境を構築している [5]．このシステムを利用し，内省行為をすることはプログラミング教育において有用であるとしている．本研究で開発したシステムでは「コンパイル」「インデント」「枠組みの導出」「実装」の評価を行うことにより，コンパイルエラー，実行エラー，論理エラーに加え，インデントの間違えの指摘，クラスの枠組みの間違えを指摘することができる．多くの項目で評価を行うことにより，学習者が内省することを容易になると考えられる．

3. アプリケーション詳細

本アプリケーションはクライアントとサーバーで構成されている．

クライアントの利用者は学習者と教授者であり，それぞれに専用のクライアントが存在している．学習者が利用する画面は提出画面であり，ファイルの提出ができる．また，提出すべきファイルのファイル名，提出したファイルのファイル名，ファイルの内容，提出時間，評価結果が表示される．教授者が利用する画面は解答状況画面であり，学習者が提出したファイルの内容や評価結果，どのファイルを提出しているかの進捗情報を確認することができる．

サーバー側のアプリーケーションは大きく分けて２つのサービスにより構築されており，全てマイクロサービスアーキテクチャを採用して開発されている．

ひとつは学習者によって提出されたファイルを管理し，教授者にファイルの提出状況等の情報を送るサービスとなっている．もうひとつは学習者によって提出されたファイルを評価するサービスとなっている．

3.1 教授者用クライアント

教授者用クラアントでは学習者の演習の進捗具合を確認できる．また，それだけではなく，演習をする上で必要な科目の作成，演習の作成，演習の際に提出するファイルの作成などもできる．利用者は授業の教員はもちろん，SA(Student Assistant) や TA(Teaching Assistant) なども対象としている．教授者クライアントでは演習全体の進捗状況と各課題ごとの進捗状況を把握することができる．

本項では各画面の詳細を記述する．

3.1.1 科目一覧画面

この画面には今までに作成した科目が科目一覧画面に全て表示される．科目の作成と今までに作成した科目の編集・削除はこの画面で行うことができる．

この画面の右下に表示されている十字の青いアイコンのボタンをクリックすることで科目の新規作成するためのダイアログを表示することができる．このダイアログに新規で作りたい科目名を入力し，作成ボタンをクリックすることで科目の新規作成をすることができる．

科目の編集は科目一覧の編集したい科目の右側にあるペンのアイコンをクリックすることでできる．ペンのアイコンをクリックすると科目編集用のダイアログが表示される．入力欄に変更したい科目名を入力し，変更ボタンをクリックすることで科目名の変更ができる．

科目の削除は科目一覧の削除したい科目の右側にあるゴミ箱のアイコンをクリックすることでできる．ゴミ箱のアイコンをクリックすると科目削除用のダイアログが表示される．ダイアログには科目の情報を削除してもよいかの確認用のメッセージが表示されている．削除しますボタンをクリックすることで科目の削除を行うことができる．

3.1.2 演習一覧画面

この画面には今までに作成した演習が演習一覧画面に全て表示される．演習の作成と今までに作成した演習の編集・削除はこの画面で行うことができる．

演習一覧画面の右下に表示されているペンの青いアイコンのボタンをクリックすることで，メニューが表示される．このメニューの十字の青いアイコンのボタンをクリックすることで演習を新規作成するためのダイアログを表示することができる．このダイアログに演習名，演習の日付，演習の時間を入力するし，作成ボタンをクリックすることで演習を新規作成することができる．

演習一覧画面の演習の右側にあるペンのアイコンをクリックすることで演習名の編集ができる．ペンのアイコンをクリックすると演習編集用のダイアログが表示される．入力欄に変更したい演習名を入力し，変更ボタンをクリックすることで演習名の変更ができる．

演習一覧画面の削除したい演習の右側にあるゴミ箱のアイコンをクリックすることで演習を削除することができる．ゴミ箱のアイコンをクリックすると演習削除用のダイアログが表示される．ダイアログには演習の情報を削除してもよいかの確認用のメッセージが表示されている．削除しますボタンをクリックすることで演習の削除を行うことができる．

3.2 解答状況画面

解答状況画面では，演習全体の解答状況と課題別の解答状況を把握するための画面がある．また，学習者によって提出されたファイルの内容と，ファイルの評価結果を閲覧することができる．

ⓒ 2020 Information Processing Society of Japan

3.2.1 全体の解答状況画面

演習全体の解答状況画面を Fig.1 に示す．この画面は上下で 2 つのエリアに分かれてる．

上部は提出されたファイル全体の提出状況を棒グラフを用いて表示されている．棒グラフの緑色がファイルを提出し評価が通っている人数，黄色がファイルを提出しているが評価が通ってない人数，赤色がファイルを提出していない人の割合を示している．

下部は各学習者の提出ファイルごとの提出状況と評価結果が，アイコンを用いて表示されている．緑色のアイコンはファイルを提出し評価が通っている状態，黄色のアイコンがファイルを提出しているが評価が通ってない状態，赤色のアイコンがファイルを提出していない状態を示している．このアイコンをクリックすることにより，後述するファイルの評価結果画面を閲覧することができる．

この画面はブラウザを更新することなく自動で，提出状況を更新するため，教授者は何もすることなく，学習者が提出することでリアルタイムに変化するのを閲覧することができる．

3.2.2 課題別の解答状況画面

課題別の解答状況画面を Fig.2 に示す．この画面は左右で 2 つのエリアに分かれてる．

左側のエリアは提出した人数と未提出の人数の割合を円グラフを，用いて表示されている．右側のエリアは提出状況をリストを用いて表示している．このリストは表示する対象を右上のタブを選択することで，提出者と未提出者の切り替えができる．提出済みのタブを選択すると，提出した順に 3 分ごとにグループ分けを行なって表示される．リストの要素にはアイコン，ID，氏名があり，アイコンによってファイルの評価結果が示されている．この要素をクリックすることにより，後述するファイルの評価結果画面を閲覧することができる．未提出のタブを選択するとその課題のファイルを提出していない学習者の一覧が表示される．

3.2.3 ファイルの評価結果画面

ファイルの評価結果画面を Fig.3 に示す．この画面では提出されたファイルが評価システムによってどのような評価を受けたか，また提出されたファイルの内容を確認することができる．Fig.3 は Java ファイルが提出された場合の画面である．この画面はダイアログで表示され，タイトルには学習者の名前とファイル名が記述されている．また，画面右上の緑色の上下左右の矢印は表示する対象を切り替えるコントローラーとなっていて，上下の矢印をクリックすることにより，学習者の切り替え，左右の矢印をクリックすることにより，課題の切り替えを行うことができる．

3.3 学習者用クライアント

学習者用クライアントの提出画面を Fig.4 に示す．学習

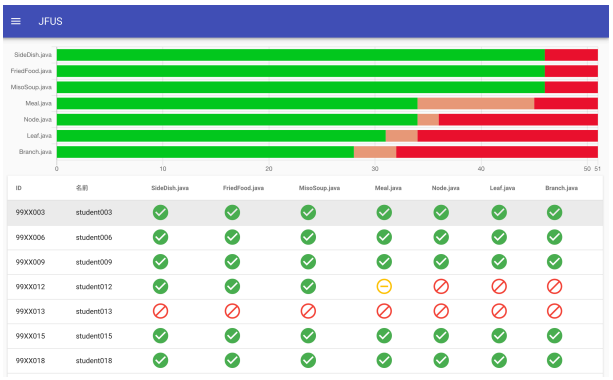

図 1 演習全体の解答状況画面
Fig. 1 Answer status screen for the entire exercise

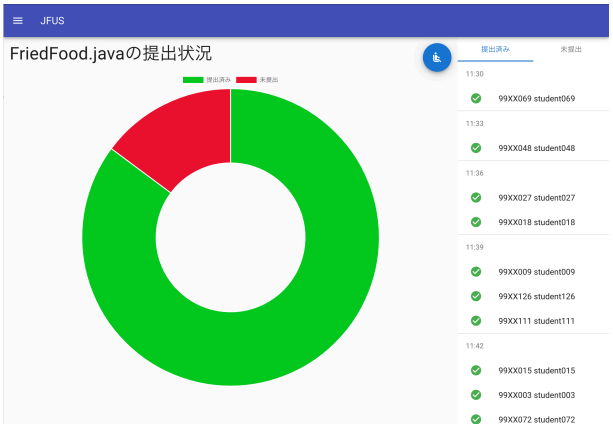

図 2 課題別の解答状況画面
Fig. 2 Answer status screen for each assignment

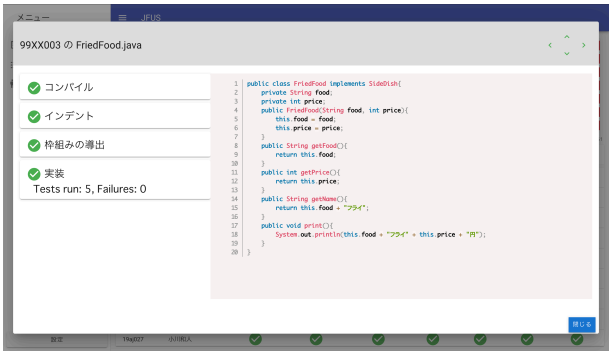

図 3 ファイルの評価結果画面
Fig. 3 File evaluation result screen

者用クライアントでは学習者が課題にあったファイルを提出することができる．提出画面では学習者が課題にあったファイルを提出することができる．提出画面の左側の課題一覧，右側のタイムラインで構成されている．課題一覧には教授者によって設定された課題が表示される．課題名の左側にそれぞれの課題の状況を示す表示がある．課題の提出状況や提出したファイルの内容によって色とアイコンが変化する．右側のタイムラインには学習者が提出したファイルが時系列で表示される．学習者は設定された課題に合わせてファイルを作成し，ファイルを提出画面にドラッグ

ⓒ 2020 Information Processing Society of Japan

図 4　提出画面
Fig. 4　Submission screen

アンドドロップすることでアップロードすることができる．アップロードされたファイルはサーバーで評価を行い，結果は課題一覧のアイコンに反映される．

3.4　Java ファイルの評価

本システムは提出されたファイルの拡張子に合わせて，そのファイルを評価する仕組みが存在している．現在，評価できる拡張子は csv と java のみとなっている．それ以外の拡張子は評価されないが，提出は可能である．

java ファイルが提出された際はコンパイル，インデント，枠組みの導出，実装の 4 項目の評価を行なっている．従来の演習では学習者は学習者自身が作成したプログラムのコンパイルと実行をする．コンパイルは文法上の間違いがないことを客観的に評価できるが，実行の場合は客観的な評価をすることが出来ない．実行結果が教授者によって提示された模範となる実行結果と同一だとしても，仕様を満たしていないことがあるかである．例えば，仕様が配列に格納されている文字列を全て標準出力に出力するであり，模範となる実行結果が標準出力に 2 個の文字列が出力されている問題があったとする．その際に，学習者が配列の中の 2 個のみを出力するプログラムを作成した場合，模範となる実行結果と同一にはなるが，本来の仕様とは違うため，間違いになる．

本システムの Java の評価機構では，コンパイル，インデント，枠組みの導出，実装を客観的に評価することができる．模範解答となる Java ファイルやそのファイルを元に作成されたテストケースを元にシステムが評価をするためである．

3.4.1　コンパイル

コンパイルの項目は，一般的な Java コンパイラと同等の評価を行う．評価システムで動的コンパイルを行い，その結果を評価結果としている．

3.4.2　インデント

インデントの項目は，インデントが正しく行われているかの評価を行う．java の場合はインデントが間違っていても問題なくコンパイルや実行を行うことができる．そのため，インデントが誤っていても，初学者の場合は発見が困難である．しかし，可読性の観点から多くのコーディング規約ではインデントを正しく行うことが求められている．そのため，Java ファイルの評価サービスでは，提出されたファイルのインデントが正しいかを確認し，誤っていた場合には，「インデントが適切ではありません」と学習者にメッセージを返す．

3.4.3　枠組みの導出

枠組みの導出とは，与えられた UML のクラス図からクラス名やメソッドおよびその引数，返却値の型など，プログラムの枠組みとなる部分を導出し，最低限，コンパイルが通る状態まで記述することをいう．クラスの形式的記述を行わせることで，コンパイルエラーが発生した際には，クラスやメソッドの宣言が不正なのか，実装部分が間違っているのかの区別が明確になる．

枠組みの導出の項目では，枠組みの導出が模範解答と完全に一致しているかの評価を行う．演習の作成時に，提出ファイルを設定することができ，その際に設定された java ファイルと提出されたファイルを比較し，正しく枠組みの導出が出来ているか確認する．枠組みの導出の評価を行う理由は 2 つある．1 つ目はコンパイルや実行はできても，正しくクラス図から枠組みの導出ができていない場合があるためである．例えば，インスタンス変数名がクラス図とは違っていても，そのインスタンス変数を呼び出す側でも間違った名前で呼び出した場合，コンパイルは正常に行え，実行もすることができる．そういった間違いを検知するために評価を行う．2 つ目はユニットテストを実行するためである．ソースコードを穴埋めではなく一から作成する場合，変数名のタイプミス，メソッドの書き忘れ，メソッドの引数が足りない等の誤りが発生する可能性が高い．その場合，ユニットテストを実行することができなくなってしまう．そのため，メソッド名等のシグネチャを確認してからユニットテストの評価を行う．

3.4.4　実装

実装の項目では，ユニットテストを用いて評価を行う．ユニットテストは JUnit を用いている．模範解答を元に事前にテストケースを作成し，評価サービスにアップロードすることにより実装の評価ができるようになる．評価の結果は，総テスト件数と誤ったテスト件数がメッセージとして返される．また，実装の評価をしたくない場合や interface 等のユニットテストが出来ない場合は，正しいという評価を返す．

ⓒ 2020 Information Processing Society of Japan

4. 試験運用

授業の演習において本システムが正しく動作するのかの確認を目的に，3 回試験運用を行なった．本システムは同時利用者数として 60 人を想定している．試験運用では想定利用者数に近い人数である，50 人前後で行なった．また試験運用での評価システムの項目はコンパイル，インデント，枠組みの導出の 3 つであった．

4.1 実施環境

本演習の対象は東京電機大学のシステムデザイン工学部の情報システム工学実験 I に参加している学生とした．対象とした学習者は全員，東京電機大学のシステムデザイン工学部で行われているコンピュータプログラミングⅠ，コンピュータプログラミングⅡ，コンピュータプログラミングⅢを全て履修済みであり，本演習で必要な基本的なコーディング技術，インデント等のコーディング規約の把握，クラス図からの枠組みの導出については学習済みである．

実験ごとに約 50 人ずつ参加，計 147 人に参加してもらい，全員を対象とした．授業では，授業資料を配布し，資料に記述されている仕様を満たすプログラムを Java 言語で実装する演習を行なった．実験は学生にブラウザを用いて授業時間に作成したファイルをアップロードしてもらった．

4.2 実験結果

実験の参加人数は 1 回目は 50 人，2 回目は 48 人，3 回目は 49 人だった．想定利用人数と同程度の人数が，同時に本システムを利用しても基本的な動作には問題がなかった．

1 回目の実験において，学習者のファイルの提出や学習者に対するフィードバックは問題なかった．また，教授者に対するフィードバックやリアルタイムに変動する項目も正しく動作することが確認できた．本システムは，ファイルを指定した順番に提出されることを想定していた．しかし，実験の際に順番通りに提出しない学生が見受けられた．それにより，依存関係があるファイルに関して，正常に評価することが出来ないことがあった．

2 回目の実験において，正しく実装されたファイルであっても，本システムが誤った評価をすることがわかった．そこで，1 回目の実験のログを確認したところ，同様の問題が起きていたことがわかった．本来であれば正解となるファイルであったが，枠組みの導出の項目において誤っていると評価されていた．誤った評価をする原因として同時に複数のファイルが提出された際に，ファイルを正しく読み込むことができず，提出したファイルと内容が違ったものを評価していることであった．このような問題が起きた理由として，同時にファイルをアップロードすることにより，過剰な負荷がかかったことであると推測された．そのため，過剰な負荷の分散を目的に，評価サービスをスケールアウトする対策を 3 回目の実験までに施した．

3 回目の実験では全てのファイルに対して，正しい評価をすることができた．評価サービスをスケールアウトしたことにより，1 つ 1 つのサービスに負荷がかかることがなくなったため，誤った評価することはなくなった．

5. 本実験

学習者が演習の場面において，どのような間違いをし，その修正にどれだけの時間を要しているかの調査を目的にシステムを利用する演習を 2 回実施した．本演習はシステムデザイン工学部のコンピュータプログラミングⅢで行った．実験ではコンピュータプログラミングⅢを履修してる学生を対象とした．演習では，授業資料を配布し，資料に記述されている仕様を満たすプログラムを Java 言語で実装を行なった．演習の制限時間は 1 時間であり，学習者には課題が完成したと思ったら随時，ファイルをシステムに提出してもらった．本システムでファイルの提出履歴と評価結果を分析することにより，学習者がどのような間違いをしたか，またその間違いをどう修正したからを調査する．

5.1 実施環境

本演習の対象は東京電機大学のシステムデザイン工学部のコンピュータプログラミングⅢに参加している学生とした．対象とした学習者は全員，東京電機大学のシステムデザイン工学部で行われているコンピュータプログラミングⅠ，コンピュータプログラミングⅡを履修済みであり，本演習で必要な基本的なコーディング技術，インデント等のコーディング規約の把握，クラス図からの枠組みの導出については学習済みである．

5.2 実験 1

実験 1 は 2019 年 11 月 7 日に東京電機大学のシステムデザイン工学部のコンピュータプログラミングⅢの講義で実施した．実験 1 には 43 名参加した．

実験 1 で提出すべきファイルは枠組みの導出のみを行う PotatoChips.java，コンストラクタと Getter を実装する Gum.java，いくつかの振る舞いを実装する Rucksack.java の 3 つだった．

5.2.1 最後に提出されたファイルの評価結果

最終的な提出において，各課題のコンパイル，インデント，機械的導出，実装の評価で誤りだと判定された個数を Table 1 に示す．本システムで利用している評価システムはコンパイルの評価が誤りだった場合，他の評価項目もすべて誤りだと判定する．また，コンパイルの評価が正しく，機械的導出の評価が誤りの場合は実装の評価は誤りとなる．そのため Table 1 では，学生がどこの項目でつまずいてい

ⓒ 2020 Information Processing Society of Japan

表 1　実験 1 の評価結果
Table 1　Evaluation result of Experiment 1

ファイル	コンパイル	インデント	枠組みの導出	実装
PotatoChips	0	0	1	2
Gum	0	0	1	2
Rucksack	6	0	0	8

るのかを明確に提示するために，インデントは評価システムが誤りとしたインデントの数から評価システムが誤りとしたコンパイルの数を引いた値，枠組みの導出は評価システムが誤りとした枠組みの導出の数から評価システムが誤りとしたコンパイルの数を引いた値，実装は評価システムが誤りとした枠組みの導出の数から評価システムが誤りとした実装の数をを引いた値となっている．平均提出回数は PotatoChips では約 1.6 回，Gum では約 1.3 回，Rucksack では約 1.7 回であった．また，演習中に再提出を行った学生は 24 人いた．

5.2.2　修正に要した時間

本システムでは提出されたファイル自体，ファイルの更新日時，ファイルの評価結果を全て履歴としてデータを収集している．そのため，ファイルの履歴を確認することにより，学習者がどのようにファイルを修正し，そのファイルがどのような評価をされているかを確認することができる．300 秒ごとに区切った修正に要した時間の分布を Fig. 5 に示す．修正に要した時間は，最初にファイルを提出した時間からファイルが全ての評価項目において正しいと判定されたまでの時間のことである．8 割以上は 20 分以内に修正が完了している．しかし，6 名の学習者は 20 分以上修正に時間がかかったことがわかる．この 6 名の学習者がどのファイルのどの評価でどれだけ時間がかかったかを Table 2 に示す．どのような間違いの修正に時間がかかっているのかの調査をするために，これらの学習者に関してより詳細に分析を行う．

5.2.3　学習者 A

学習者 A は PotatoChips の評価において，誤りと判定された項目は枠組みの導出であり，学習者に対してのフィードバックのメッセージは「状態の宣言が違います」であった．提出されたファイルを確認したところ，ファイルは毎回修正されており，試行錯誤をしたことが伺える．最初に提出されたファイルを確認したところ，枠組みの導出が誤りと判定された理由はインスタンス変数の初期化が間違っていた為であった．PotatoChips はインスタンス変数が 2 個あり，最初の提出ではその 2 個とも初期化が間違っていた．修正を繰り返す中で，3 回目の提出において，1 個の初期化を修正し，9 回目の提出でもう 1 個の初期化も修正することができた．最初に提出されてから正解になるまでにかかった修正の時間は，30 分であった．

5.2.4　学習者 B

学習者 B は Rucksack の評価において，誤りと判定された項目は実装であった．提出されたファイルを確認したところ，枠組みの導出はしていたものの実装部を書いていなかった．これは，学習者自身が完成していないのを認識していながら，枠組みの導出までに間違いがないかを確認するために，行なった提出だと考えられる．この学習者は枠組みの導出をシステムにより間違っていないことを確認し，その後，実装を行い提出している．

5.2.5　学習者 C

学習者 C は PotatoChips の評価において，誤りと判定された項目は実装であった．提出されたファイルを確認したところ，2 回目に提出されたファイルは 1 回目に提出されたファイルと全く同じ内容であり，評価が誤りだった原因は，返却値が空文字だったためである．PotatoChips は枠組みの導出のみを行う課題なため，返却値の型が参照型の場合は返却値を null にしなければならない．また，他の課題の提出時間を確認したところ，学習者 C は PotatoChips が正解してないにも関わらず，次の課題に進んでいたことがわかった．これは，なにが原因で評価が誤りとなっているのかがわからなかった為だと考えられる．他の課題をやっていた時間も含め，最初に提出されてから正解になるまでにかかった修正の時間は 55 分であった．

5.2.6　学習者 D

学習者 D は PotatoChips の評価において，誤りと判定された項目は枠組みの導出であり，学習者に対してのフィードバックのメッセージは「振る舞いの宣言が違います」であった．評価が誤りだった原因は，インスタンスメソッドであるメソッドに static 句がついていたためであった．学習者 C 同様，学習者 D は PotatoChips が正解してないにも関わらず，次の課題に進んでいたことがわかった．提出されたファイルの履歴を確認したところ，PotatoChips，Gum，PotatoChips の順に提出されており，提出の時間のそれぞれの差は 20 分と 27 分であった．最初に提出されてから正解になるまでにかかった修正の時間は 46 分であり，他の課題をやっていた時間を除いても，20 分以上時間を要したと考えられる．

5.2.7　学習者 E

学習者 E は Rucksack の評価において，誤りと判定された項目は実装であった．提出されたファイルを確認したところ，枠組みの導出はしていたものの実装部を書いていなかった．これは，学習者 B 同様，学習者自身が完成していないのを認識していながら，枠組みの導出までに間違いがないかを確認するために，行なった提出だと考えられる．この学習者は枠組みの導出をシステムにより間違っていないことを確認し，その後，実装を行い提出している．

ⓒ 2020 Information Processing Society of Japan

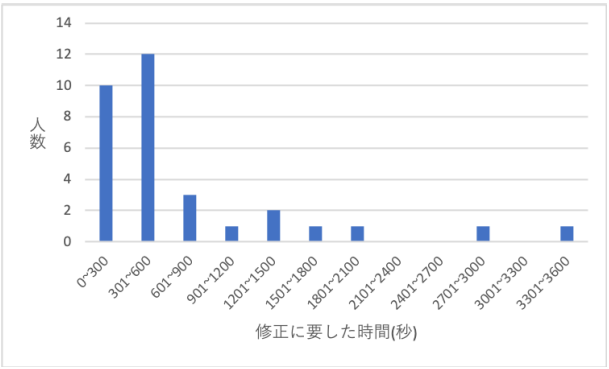

図 5　実験 1 において修正に要した時間の分布
Fig. 5　Distribution of correction time for experiment 1

表 2　実験 1 の修正時間
Table 2　Experiment 1 correction time

学習者	ファイル	評価項目	修正時間 (秒)
学習者 A	PotatoChips	枠組みの導出	1837
学習者 B	Rucksack	実装	1453
学習者 C	PotatoChips	実装	3306
学習者 D	PotatoChips	枠組みの導出	2790
学習者 E	Rucksack	実装	1697
学習者 F	Rucksack	枠組みの導出	1453

5.2.8　学習者 F

学習者 E は Rucksack の評価において，誤りと判定された項目は枠組みの導出であった．最初に提出されたファイルを確認したところ，メソッド名が間違っていたため，枠組みの導出の項目において誤りと判定されていた．2 回目の提出においてこの間違いは修正されていた．ただし，メソッド名の間違いに気づくのに 24 分かかったわけではなく，メソッド名の修正と実装部の実装をしていたため，2 回目の提出までに時間がかかったと考えられる．1 回目の提出ではメソッド名が間違っていただけではなく，実装部も書いていなかった．そのため，メソッド名が正しかったとしても評価が通ることはなかった．学習者 B と同様に，枠組みの導出までに間違いがないかを確認するために，行なった提出だと考えられる．

5.3　実験 2

実験 2 は 2019 年 11 月 11 日に東京電機大学のシステムデザイン工学部のコンピュータプログラミングⅢの講義で実施した．実験 2 には 42 名参加した．

実験 2 で提出すべきファイルは枠組みの導出のみを行う StrawberryFrame.java，コンストラクタと Getter を実装する Strawberry.java，いくつかの振る舞いを実装する Plastictray.java の 3 つだった．

5.3.1　最後に提出されたファイルの評価結果

最終的な提出において，各課題のコンパイル，インデント，機械的導出，実装の評価で誤りだと判定された個数を

表 3　実験 2 の評価結果
Table 3　Evaluation result of Experiment 2

ファイル	コンパイル	インデント	枠組みの導出	実装
StrawberryFrame	1	1	0	0
Strawberry	0	0	0	0
PlasticTray	10	1	3	6

Table 3 に示す．平均提出回数は StrawberryFrame では約 1.4 回，Strawberry では約 1.4 回，PlasticTray では約 1.7 回であった．また，演習中に再提出を行った学生は 25 人いた．

5.3.2　修正に要した時間

300 秒ごとに区切った修正に要した時間の分布を Fig. 6 に示す．修正に要した時間は最初にファイルを提出した時間からファイルが全ての評価項目において正しいと判定されたまでの時間のことである．8 割以上は 20 分以内に修正が完了している．しかし，5 名の学習者は 20 分以上修正に時間がかかったことがわかる．5 名の学習者がどのファイルのどの評価でどれだけ時間がかかったかを Table 4 に示す．どのような間違いの修正に時間がかかっているのかの調査をするために，これらの学習者に関してより詳細に分析を行う．

5.3.3　学習者 G・H・I・J

学習者 G・H・I は 2 回提出しており，1 回目の提出では枠組みの導出のみを行なったファイルであった．学習者 J は 5 回提出しているが，最初の 3 回目までは前述の学習者と同様に枠組みの導出のみを行なったファイルだった．これは，学習者自身が完成していないのを認識していながら，枠組みの導出までに間違いがないかを確認するために，行なった提出だと考えられる．この学習者らは枠組みの導出をシステムにより間違っていないことを確認し，その後，実装を行い提出している．

5.3.4　学習者 K

学習者 K は PlasticTray の評価において，誤りと判定された項目は枠組みの導出であり，学習者に対してのフィードバックのメッセージは「状態の宣言が違います」であった．評価が誤りだった原因は，インスタンス変数にアクセス修飾子の「private」がなかったためである．提出されたファイルの内容を確認したところ，3 回目の提出時点で実装は完成していた．それ以降の 4 回目から 6 回目まではメソッドの実装部を修正して，提出をしていた．3 回目の提出から 7 回目の提出までにかかった時間は約 14 分であった．

6.　考察

本システムでは，提出されたファイルと評価結果等が履歴として情報が収集されている．それにより，演習中，または演習終了後に学習者が 1 つの課題にどれだけ時間がか

ⓒ 2020 Information Processing Society of Japan

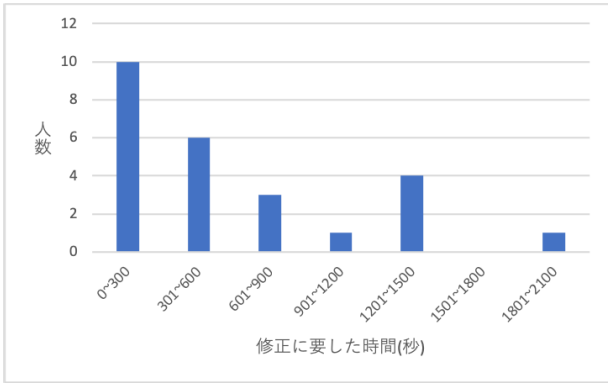

図 6　実験 2 において修正に要した時間の分布
Fig. 6　Distribution of correction time for experiment 2

表 4　実験 2 の修正時間
Table 4　Experiment 2 correction time

学習者	ファイル	評価項目	修正時間 (秒)
学習者 G	PlasticTray	実装	1869
学習者 H	PlasticTray	実装	1451
学習者 I	PlasticTray	実装	1489
学習者 J	PlasticTray	実装	1446
学習者 K	PlasticTray	枠組みの導出	1253

かったか，どのような間違いをしていたか等の分析を行える．また，従来の演習では多くの場合，最後の提出のみしか確認しないため，どの項目で学習者が躓いているかを把握するのが困難である．本システムが収集した情報を分析によって最後の提出からでは読み取ることができない間違いを発見できるではないかと考えられる．また，システムからのフィードバックを元にどのような修正を行なっているかの試行錯誤が詳細を読み取れることが確認できた．

修正に多くの時間を要した学生の提出されたファイルを分析したところ，枠組みの導出において修正に時間がかかることがわかった．これは枠組みの導出が誤っていてもコンパイラーでは指摘されない上，仕様を満たす動作をする場合があるため，間違いに気づくことが困難である．また，本システムでは「状態の宣言が違います」や「振る舞いの宣言が違います」等の間違いを指摘するフィードバックはするが，間違いがある行は指摘しない．そのため，間違いがあることに気付けても，間違いをしている箇所を特定し，それを正しく修正するのに試行錯誤をするため，多くの時間が必要になる学習者がいると考えられる．

学習者の中には誤りがあることを認識していながら，ファイルを提出している学習者がいた．今回行った実験では，学習者に対し，１つ１つの課題を完了したら，ファイルを提出するように指示をしていた．この学習者らは，誤りがあることを認識していながら，それらを検証させるためにシステムに提出を行っていたと考えられる．システムに提出することにより，自身では検証できないインデントや枠組みの導出，実装の評価結果をフィードバックしてもらえる．学習者は課題が完成する前の段階でシステムによって評価してもらうことにより，現状の段階までで間違いがないかを確認していると考えられる．このような提出は学習者にとって，段階を踏みながら課題を進められるため，課題を解く上で有用に働くと考えられる．ただし，教授者はこのような提出あることを把握した上で学習者の提出状況を確認する必要がある．

本システムにはファイルの提出とファイルの評価結果が履歴があり，そこから評価の推移を示すことと評価結果のより詳細な分類分けを可能である．それらの傾向から，「修正に多くの時間がかかる学習者」の前兆を予測することが可能であると考えられる．修正に多くの時間がかかる学習者の情報を蓄積し分析することにより，そのような学習者の早期発見をすることができるようになると考えられる．

7. まとめ

本研究では学習者が課題を提出し，教授者がリアルタイムで学習者全体の学習進度を把握するシステムを開発している．ファイルを学習者にアップロードしてもらうことでファイルを評価し，評価結果を含めた情報を教授者に提示することができる．学習者は自分自身だけでは気づくことができない間違いをシステムを発見し，それを元に多くの学習者はその間違いを修正することができる．その為，教授者は課題を解くことができない学生に，より注力して指導することができる．また，本システムを教授者が利用することで机間巡視することなく，学習者の学習進度と指導を必要としている学習者を把握することができる．それにより，机間巡視ができない授業でも学習者にあわせた授業速度で授業を行うことができるため，授業の最適化が期待できる．

参考文献

[1] 堀口悟史，井垣 宏，井上亮文，山田 誠，星 徹，岡田 謙一：講義資料閲覧ログを用いたプログラミング講義進捗管理手法の提案，情報処理学会論文誌，Vol.53, No.1, pp.61-71 (2012)
[2] 加藤利康，石川 孝：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，Vol.55, No.8 , pp.1918-1930 (2014)
[3] 西田知博，原田 章，中村亮太，宮本友介，松浦敏雄:初学者用プログラミング学習環境 PEN の実装と評価，情報処理学会論文誌，Vol.48, No.8, pp.2736–2747 (2007)
[4] 斐品正照，徳岡健一，河村一樹:構造化チャートを用いたアルゴリズム学習支援システム，情報処理学会論文誌，Vol.45, No.10, pp.2454–2467 (2004)
[5] 知見邦彦，櫨山淳雄，宮寺庸造:失敗知識を利用したプログラミング学習環境の構築，電子情報通信学会論文誌 D-I，情報・システム，I-情報処理，Vol.88, No.1, pp.66–75 (2005)

ⓒ 2020 Information Processing Society of Japan