修士論文

多人数でのプログラミング演習における学習者のコーディング状況の把握システムに関する研究

平成 27 年度修了

三重大学大学院工学研究科

博士前期課程　電気電子工学専攻

彦坂　知行

目次

1 章　はじめに

プログラミングの授業では，一般に，プログラム作成を行う演習が行われている．しかし，多人数の場合，プログラムの作成が円滑に進まず指導を必要としている学習者を机間巡回で見つけることは困難である．著者らは，プログラム作成の演習中に，学習者の状況をリアルタイムで把握することができるプログラミング演習システム PROPEL（PROgramming Practice Easy for Learners）を開発・運用している[1][2][3]．このシステムでは講師が iPad 等のタブレット端末上に表示した座席表の画面に表示された情報から，指導が必要と思われる一定時間以上プログラムを変更していない学習者を見つけることができる．

従来の PROPEL では，学習者が一定時間プログラムをまったく変更しない場合しか検知することしかできておらず，指導が必要な学習者の一部しか講師はシステムを介して把握することができなかった．その結果，ある課題で動作に問題のあるプログラムが 40 個中 31 個提出されており，十分な指導を行えていない学習者が存在すると考えられる．また，4 章で述べるように，他の先行研究においてもこれらの指導が必要な学習者を検出できるシステムはないと考えられる．

本研究では，コンパイルエラー，動作エラー，スタイルの間違いについて，長時間気づいていない，あるいは直せない学習者に指導を行う必要があると考えた．本研究の目的はこれらの指導が必要な学習者に指導するために，必要な情報を講師に提供することである．

提案するシステムではコンパイルエラーが長時間あるいは多数残っているような学習者，コンパイル成功後のプログラムが課題の要求に対して正しく動作しないことに気づいていない学習者，または，そのスタイルが適切でない学習者のリストを演習中に講師に提供する．

本論文の構成は以下のとおりである．2 章で本研究での指導の方針について，3 章ではプログラミング演習システムについて述べる．4 章で先行研究の問題点について述べ，5 章で提案する機能について述べる．6 章では提案するシステムについて述べ，7 章で運用結果について述べ，8 章で本論文のまとめを述べる．

2 章　本研究における指導の方針

PROPEL はプログラミング演習を効率的に進めることを目的にしている．そのため, Web ページ上の単一の環境で演習に必要な作業を行えるように，学習者が長時間編集していないことを講師に表示したりするような支援を行っている．

PROPEL は主にプログラム初心者向けのクラスを想定している．このクラスにはプログラム作成を全く進めることのできない学習者や，コンパイルエラーが直せない学習者，動作の間違いがあるプログラムを提出してしまう学習者などがおり，これらの学習者には指摘が必要だと考えられる．

学習者への指摘を，システムが行うか，講師が行うかどうかについての方針について述べる．システムが指摘でき，かつ，学習者自身で修正できると思われる場合はシステムが指摘を行う．システムが指摘できないあるいは学習者自身で修正できないと思われる場合は講師が指摘を行う．本来は学習者自身で気づき，修正を行うべきだが，効率的に演習を進めるために上記の方針とした．

プログラミングの教育支援では，様々なコーディング支援機能を使用することも考えられる．しかし学習者にプログラムを自力で作成できる能力を身に着けさせるために，支援をどの程度行うかは検討する必要がある．例えば，Visual Studio[4]，Eclipse[5]などソフトウェア統合開発環境ではコーディングの効率を上げるために，開き括弧を入力したらそれに対応する閉じ括弧を自動で入力する，字下げを自動的に調整するなど，様々なコーディングの支援が行われている．プログラミング演習システムでも同様な支援を行うことが考えられる．しかし，学習者がシステムの補助なしでプログラムを作成できないのは問題である．そこで我々が開発中の PROPEL においては学習者がシステムの補助に頼り切ることがないようにすることを目指している．具体的には学習者がプログラムを提出して変更ができないようになってから，動作テストやスタイルチェックの結果を返すようにして提出前に自力で動作やスタイルのチェックを行うよう促している．スタイルの間違いについては，次の課題で修正すればよいので学習者には再提出はさせない．動作の間違いに関しては課題ごとに違うため，結果を返

し，プログラムを修正後に再提出させる．

初歩的なエラーについては学習者がプログラムを編集中に，学習者に提供する．このエラーについては簡単な間違いであるので，学習者が様々な課題でコーディングを行うことで間違いが減っていくと考えられるため，また，講師の負担を軽減し，演習を効率的に行うため，システムにより指摘を行う．

図 1 に PROPEL と IDE（統合開発環境）の支援機能の比較を示す. PROPEL では IDE が提供できていないスタイルの間違いについての情報を提供している．

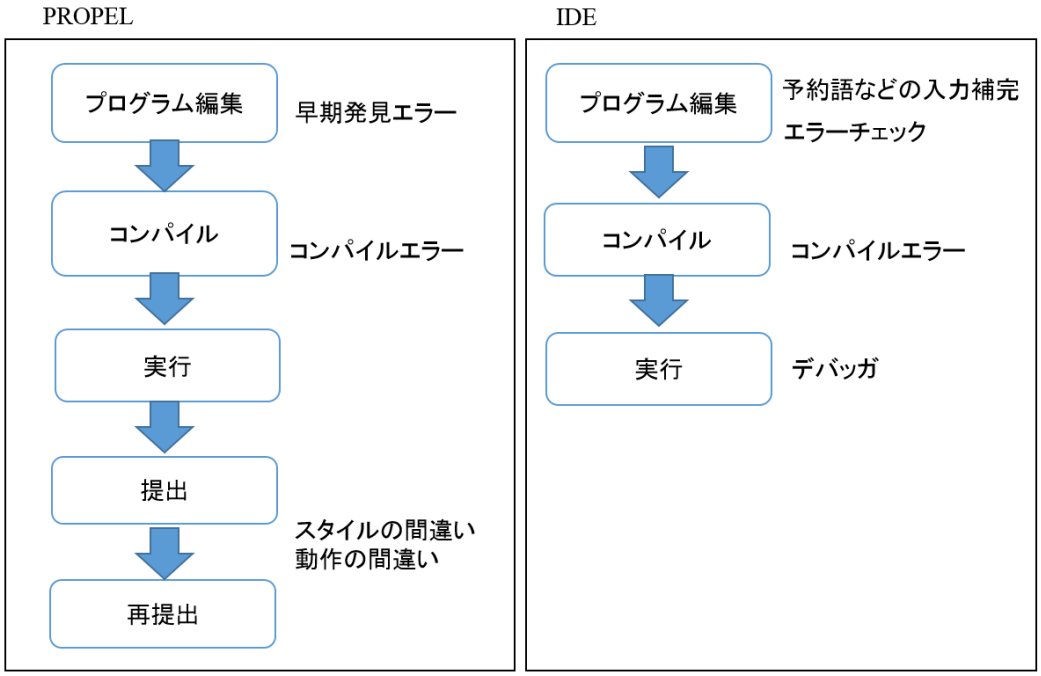

図 1　PROPEL と IDE の支援機能の比較

3 章　従来のプログラミング演習支援システム

図 2 に従来のプログラミング演習システム PROPEL の構成図を示す．学生はプログラミング演習において必要な作業のうち「デバッグ」以外の「コーディング」「保存」「コンパイル」「プログラムの実行」「提出」を Web 上のひとつの画面上で行うことができる．また，受講者が講師を呼び出すことができる「呼び出し」機能を提供している．学科の科目で教えているプログラミング言語が C 言語であったので, 従来の PROPEL も C 言語を対象としている．

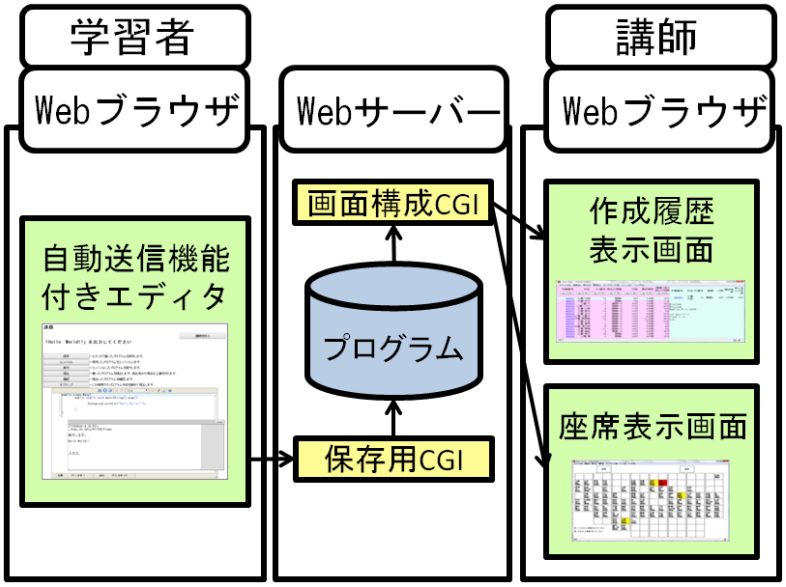

図 2　従来の PROPEL のシステム構成図

学習者は Web ブラウザを使用し図 3 に示す画面でプログラムの編集・保存・コンパイル・実行・提出を行う．保存・提出したプログラムは Web サーバ上に保存される．また，編集中のプログラムも講師に提供する情報を作成するために, 学習者には通知せずに 30 秒ごとに自動的に保存している．

図 3　プログラムの編集画面

講師は，図 4 に示す講師用画面を閲覧することで，クラス全体の進行状況を把握することができる．受講者が「呼び出し」機能を用いた場合には，座席が赤色で表示される．また，受講者がコーディング途中のプログラムが 5 分以上同じ状態，すなわち入力や修正が全く行われていない場合には，黄色で，さらに 15 分以上だと橙色で表示される．これにより，講師はプログラムの作成に行き詰っている受講者のところへ指導しにいくことができる．

講師は，また，図 5 に示す受講者の一覧を表示した講師用画面を閲覧することでも，クラス全体のプログラミング演習の進行状況を把握することができる．受講者を指定すると，この画面の右側の部分にその受講者が現在コーディング中のプログラムが表示される．これにより，気になる受講者のプログラムの現在の様子を知ることができる．

図 4　従来の講師用画面：演習室内座席表

図 5　受講者一覧，および，ある受講者のプログラム

プログラミング演習システム PROPEL は，図 6 に示すように，携帯可能で無線 LAN によっていつでもシステムにアクセスすることのできる情報端末 iPad 用の講師用画面を提供している[6]．iPad 等を利用することで，講師は教室内を巡回中でもシステムの利用を通じてプログラミング演習の進行状況を知ることができる．

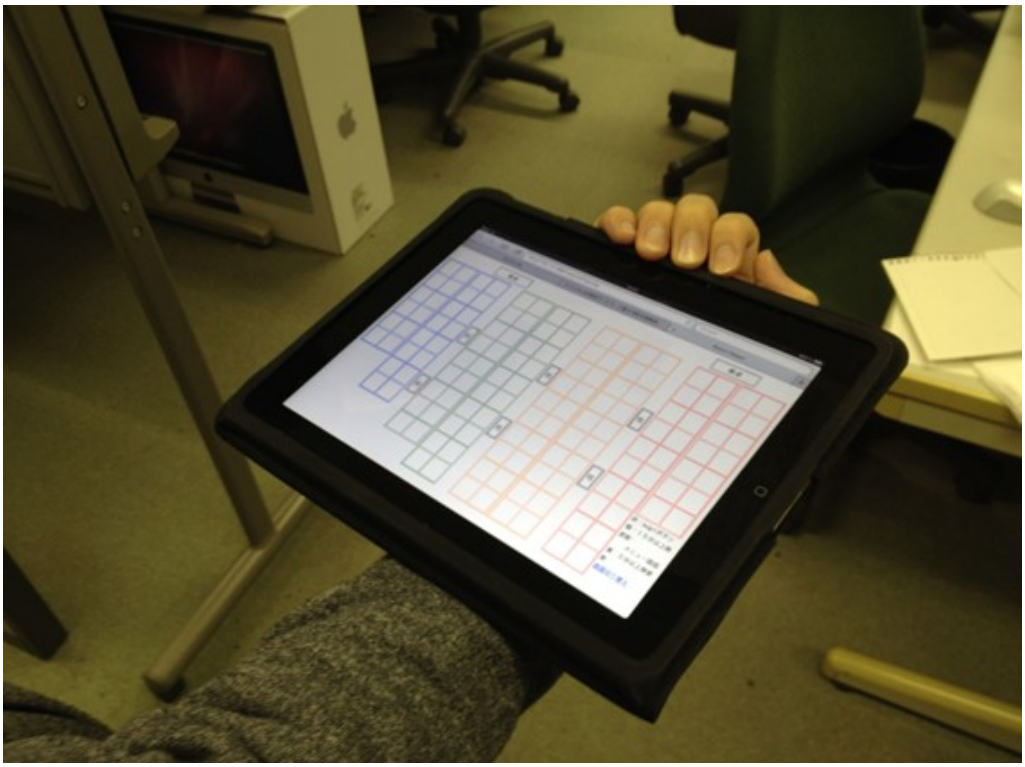

図 6　iPad での PROPEL の利用画面

4 章　先行研究の問題点

4.1 従来の PROPEL の問題点

従来の PROPEL の問題点は 2 つある．

1 つ目は一定時間入力を行っていない学習者しか検出できないため，長時間エラーが直せず試行錯誤している学習者を見つけられないことである．

2 つ目は学習者からの呼出しや，学習者が一定時間入力を行っていないことを元に指導を行うため，演習時間の早い段階では，指導が行えず，演習時間の後半に集中してしまい，効率的な指導が行えないことである．図 7 にある日の演習での指導した時刻を示すタイムチャートを示す．演習が始まってから，ある程度時間がたってからの指導回数が多いことがわかる．

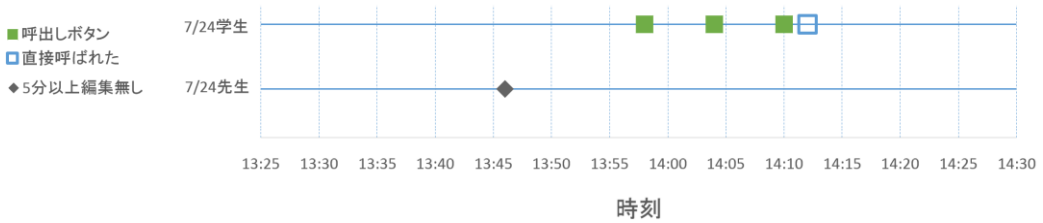

図 7　指導時刻のタイムチャート

また，3 章図 5 で示した学習者の一覧とプログラムを表示する講師用の画面は多くの学習者のプログラムを閲覧するのには時間がかかるため，演習中に講師は利用していなかった．

4.2　その他の先行研究の問題点

プログラミング演習中の状況把握に関する研究としては以下の研究がある．それぞれについて概要と問題点を説明する．

参考文献[7]で加藤らは，学習者の作業の進度集計，エラーの分類集計，作業の遅れている学習者の検出等の機能を持つ学習状況を把握するシステムを提案した．作業の進度集計とは編集開始，コンパイル，実行，正解，提出の各作業の人数を表示できる機能である．エラーの分類集計機能とは発生したコンパイルエラーメッセージを分類・集計し，講師がクラス全体で発生しているエラーの内容を把握できるようにする機能である．作業の

遅れている学習者の検出機能とは学習者の作業時間に外れ値の分析を行い，クラス全体と比べて作業が遅れている学習者を検出･表示する機能である．

上記の加藤らのシステムは座席表が表示されないため，講師が指導を行うとき手間がかかると考えられる．また，加藤らのシステムでは学生がコンパイルを行うまで間違いをチェックすることができないため，それまで状況を把握できない問題点がある．

参考文献[8]で倉澤らは学生のコンパイルエラーメッセージ履歴から，プログラムの動作理解が困難な原因と場所を推定・集計するシステムを提案した. このシステムではコンパイルエラーメッセージを過去 2 年分収集し，それぞれにエラータイプという 257 種類の ID を割り当てた. エラーの要因を分析し，学習者のコンパイル時にエラーメッセージに対して過去出現頻度の高いエラー要因を表示する．

倉澤らのシステムも座席表が表示されないため，講師が指導を行うとき手間がかかると考えられる．また，学生がコンパイルを行うまで間違いをチェックすることができないため，それまで状況を把握できない問題点がある．

5 章　提案する機能

2 章で述べたように，本システムでは主にプログラミング初心者がいるクラスを対象としている．このようなクラスでは指導が必要な学習者としては次のような学習者が考えられる．

- コンパイルエラーが直せず，困っている学習者．
- 動作エラーが直せないで困っている，あるいは気づかない学習者．
- プログラミングスタイルが直せないで困っている，あるいは気づかない学習者．

講師がこれらの指導が必要な学習者に指導できるようにするために，4 種類の情報を講師に提供する[9]．

- コンパイルエラー
- 早期発見エラー
- 動作エラー
- スタイルの間違い

以下でそれぞれについて説明する．

5.1　コンパイルエラー

学習者がコンパイルしたときにシステムでコンパイルエラーのメッセージを記録し，コンパイルエラーをなくせない学習者のリストを講師に提供する．

5.2　早期発見エラー

演習ではプログラムの間違いを早期に発見し，早期に修正することが重要であると考えられる．演習の方針では学習者はなるべく自力でエラーを修正するべきだとした．しかし，効率的に演習を進めるために軽微なエラーについてはシステムによって繰り返し指摘して学習者に修正をさせることで学習者が自力でエラーを修正できるようにする．コンパイルエラーの情報は，学習者がコンパイルするまで得ることができない．小島は初心者が起こしやすいエラーについて調査し，プログラムが完成する前でも，1 行単位で間違いがないかどうかチェックすることのできる方法を提案した

[10]．この方法によりプログラム作成中でも間違いを検出し，早期に学習者に間違いをシステムが表示する．チェックするエラーを表 2 に示す．

小島の提案では早期エラーチェックの情報は学習者に対して表示するが，本研究では，それに加えて講師に対して情報を表示することで指導すべき受講者を早期に発見し，指導を行えるようにする．

表 2　早期発見エラーの種類

セミコロン忘れ	文末に「；」を付けるのを忘れた場合のエラー
全角文字の混入	全角の文字が入っていた場合に表示されるエラー
ダブルコーテーションの閉じ忘れ	「”」を閉じ忘れた場合に表示されるエラー
括弧の開き忘れ	括弧を開き忘れたときに表示されるエラー

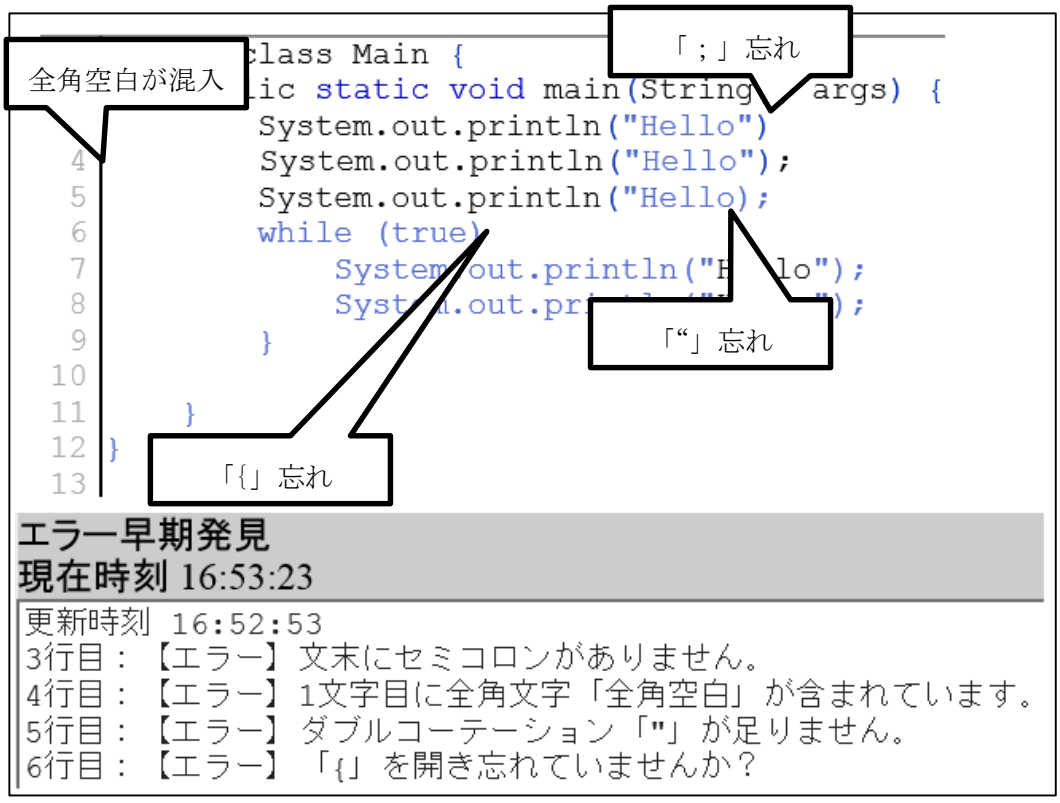

図 8　早期発見エラーの例

具体例を図 8 に示す．3 行目はセミコロンがなく，4 行目は先頭に全角空白が入っており，5 行目はダブルコーテーションを閉じ忘れており，6 行目は中括弧を開いていないのでそれぞれエラーが表示されている．

5.3　動作エラー

学習者がコンパイルエラーを直せない場合は本システムで見つけることができる．しかし，コンパイルができた場合，学習者は自分で正しく動作するかの確認を十分せずにプログラムを提出してしまうことがある．講師がこのような学習者に演習時間中に気づくことは難しいと考えられる．そこで本システムでは自動テスト機能を実装した．提出されたプログラムはシステムによって正しく動作するか自動テストを行う．自動テストは講師が設定したテストケースを満たすかどうかをテストするブラックボックステストを用いる．

自動テストの方法については検討する必要がある．

図 9 に演習で使用している教科書[11]の演習問題を示す．このような課題では出力の書式が厳密に定められていないことがある．学習者ごとに空白の入れ方などの点で出力の細かい表現は異なるため，そのままでは自動テストできない．

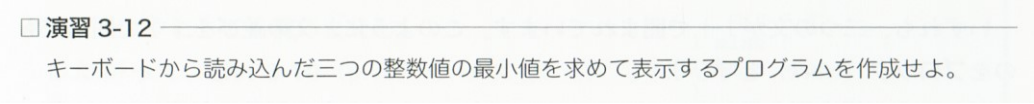

図 9　演習問題の例

キーボードから読み込んだ三つの整数値の最小値を求めて表示するプログラムを作成せよ．空白は入れず「：」は全角とすること．出力書式以外の文字列は出力しないこと．行の最後に改行を出力すること．

【出力書式】

最大値：12

図 10　出力書式を厳密に指定した演習問題の例

これを解決するためには 2 つの方法が考えられる．

1 つ目の方法は，演習問題の中で出力の書式を厳密に指定する方法である．図 9 の演習問題の出力書式を厳密に指定すると図 10 のようになる．このように書式を厳密に指定した場合，初心者には難しい，または，時間がかかると考えられる．

2 つ目の方法は，出力に指定した文字列が含まれているかどうかでテストする方法である．望月はプログラミング初心者向けに出力に関する条件を緩めたテスト[12]を提案した．この自動テストでは，ある入力に対して正解となる出力に含まれなければならない文字列を指定し，その文字列が学習者のプログラムの出力に含まれるかどうかで正誤を判定する．

例えば，入力した整数が奇数か偶数か判定するようなプログラムに対しては，入力値に「10」，正解となる出力に含まれなければならない文字列に「偶数」などと設定することでテストを行う．

望月が提案したテスト方法では．出力の形式を厳密に指定する必要がないため，出力の書式を厳密に指定するよりも，学習者が解答しやすいと考えられる．そのため本研究では望月が提案したテスト方法を用いる．

図 11 と図 12 に動作テストの画面を示す．この画面では講師が設定した複数のテストケースに対してテストを行い，結果を学習者や講師が見ることができる．例として図 7 は入力された数値の平均等を計算する，あるプログラムをテストした結果で，正しい標準偏差の値である「25.9」が含まれていないため動作失敗と判定される．これらの情報を第 6 章の図 16 のように演習中に講師に提供する．ただし実装が間に合わなかったため効果の検証は行っていない．

動作の間違いに関しては課題ごとに動作が違うため，結果を学習者に返し，プログラムを修正後に再提出させる．自動テストの結果を見ても動作エラーが長時間直せない学習者については，講師が指導を行う必要があると考えられる．

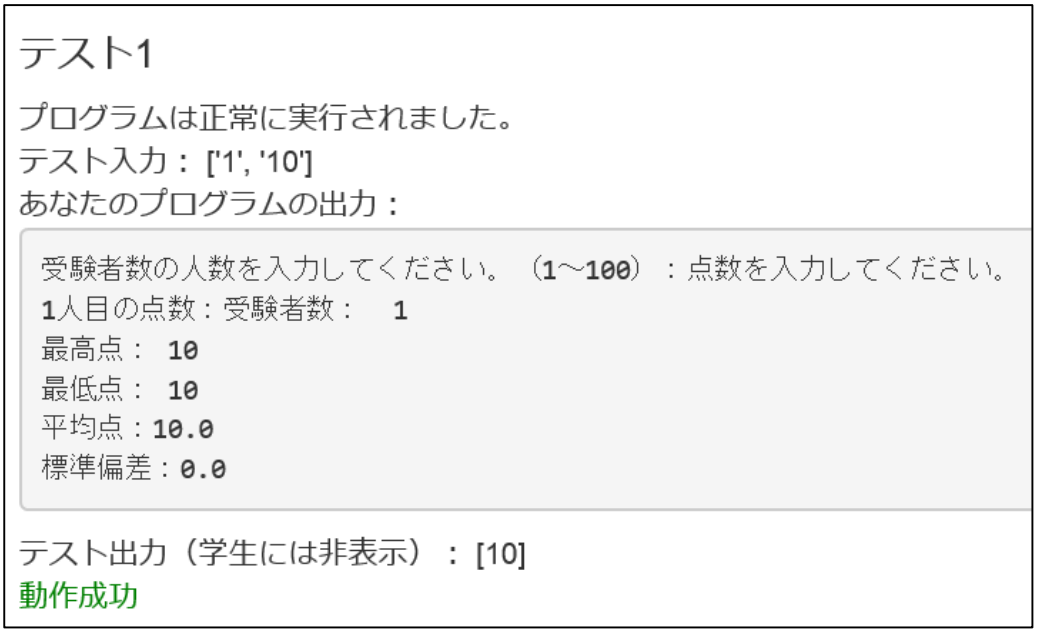

図 11　自動テストの結果（動作成功時）

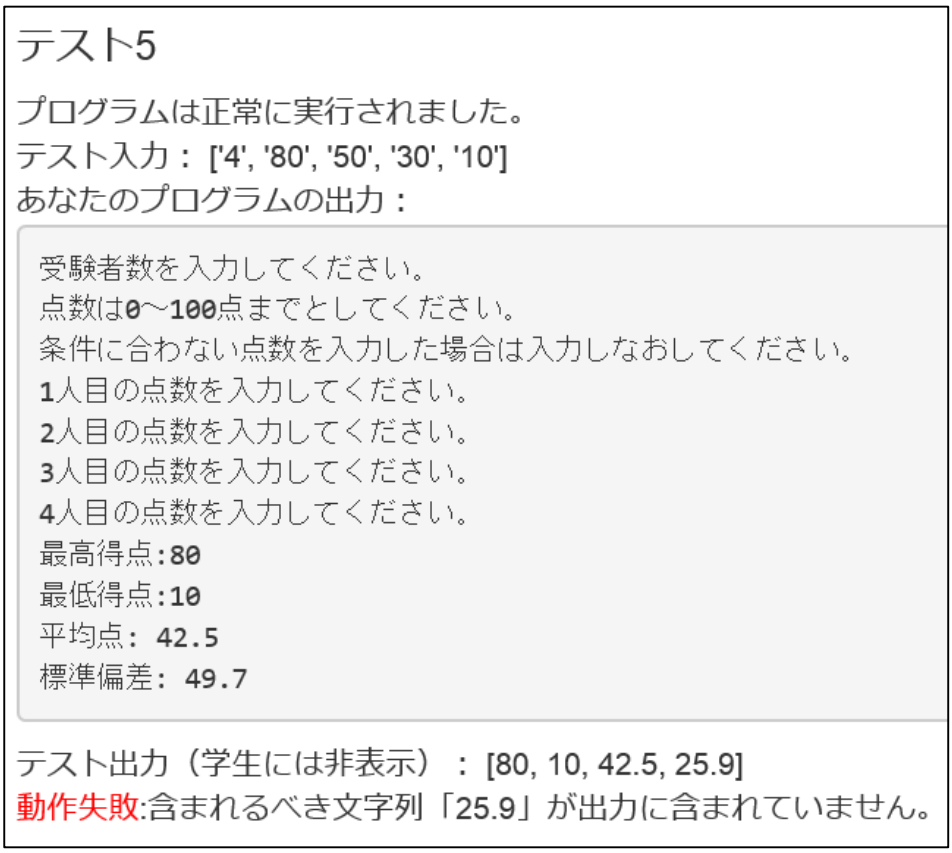

図 12　自動テストの結果（動作失敗時）

5.4　スタイルの間違い

プログラミングにおいてスタイルは重要である．プログラムの可読性を高めるためには正しいスタイルでプログラムを書く必要がある．上村はプログラムを整形するオープンソースのツールである「Artistic style」[13]を用いて，学習者のプログラムのスタイルを採点し，間違いを指摘できるシステムを作成した[14]．このシステムでは空白の入れ方や字下げの間違い等を図 13 に示すように指摘できる．14 行目では「a - b」のように演算子の前後に空白が入っていない点や，16 行目のように字下げの空白の数が間違っている点が指摘されている．本システムではこのスタイルチェックの結果を第 6 章の図 16 の画面で講師に提供し，指導する際に役立てる．ただし実装が間に合わなかったため効果の検証は行っていない．スタイルの間違いについては，次の課題で修正すればよいので学習者に再提出はさせない．

清書前

間違い行数: 5/19　　正解率: 73.7%

```
 1./*
 2.*  課題番号: #8
 3.*  プログラムの概要: 二つの自然数の差の算出
 4.*  完成日時: 2015/5/28
 5.*
 6.*
 7.*/
 8.
 9.public class Main {
10.    public static void main(String[] args) {
11.        System.out.println("二つの自然数を入力してください。");
12.        int a = new java.util.Scanner(System.in).nextInt();
13.        int b = new java.util.Scanner(System.in).nextInt();
14.        int c = a-b;
15.        if (a>=b) {
16.          System.out.println("二つの自然数の差は"+ c +"です。");
17.        } else
18.          System.out.println("二つの自然数の差は" + -c + "です。");
19.    }
20.}
21.
```

図 13　スタイルチェックの結果

6 章　提案するシステム

提案するシステムの構成図を図 14 に示す. 指導に必要な情報を講師に提供するために，本研究では講師用画面の座席表示部分について変更を行った．講師がエラーを長時間出している学習者を把握できるよう，また指導に行きやすくすることを目的に講師用画面を設計した.

提案するシステムの画面遷移図を図 15 に示す. 講師は学習者一覧画面で

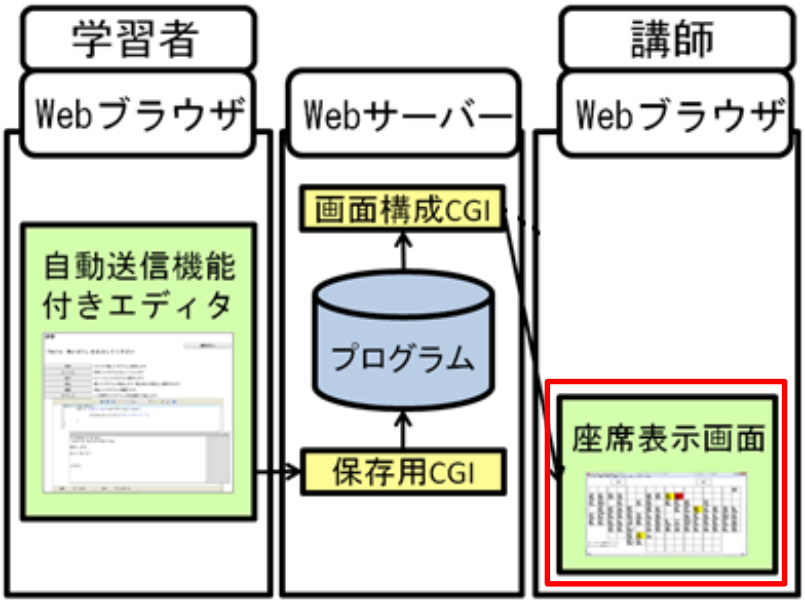

図 14　提案するシステムの構成図

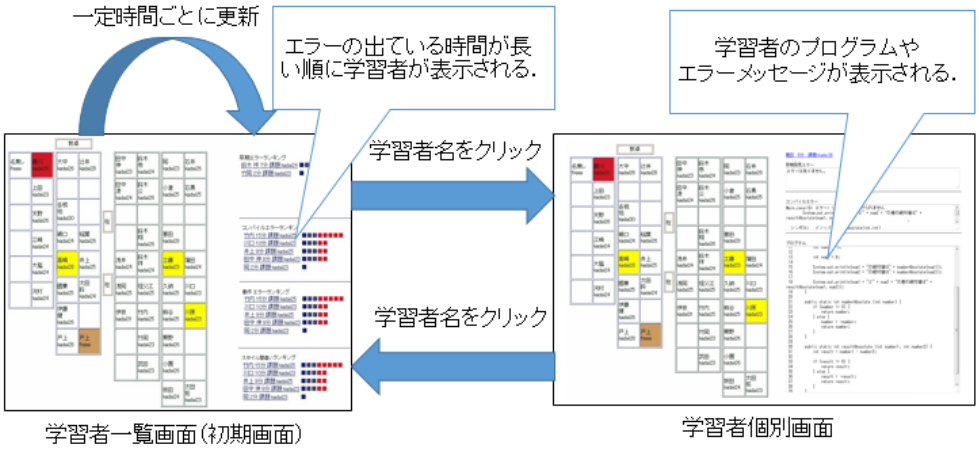

図 15　提案するシステムの画面遷移図

エラーが長時間残っている学習者の一覧を閲覧し，学習者名をクリックすると表示される学習者のプログラムやエラーの情報を表示する個別画面を閲覧し指導しに行くかどうかを判断する．

提案するシステムの画面を図 16 に示す. 画面の左側には従来の PROPEL の機能で座席表を表示する. 右側に学習者のエラーの残留時間を表示する. ここで右側の情報はエラーの残留時間が長い順番にソートされている．学生の氏名，エラーの残留時間について表示する．エラーの残留時間については棒グラフで表示し, 長くなるほど色が青から赤に変化するようにして, 講師が状況を速やかに把握できるようにした． また，エラーの残留時間が一定時間以下，今回は 2 分以下の学習者は表示しないことで，講師に不要な情報が表示されないよう情報を絞り込んでいる．

学習者の名前をクリックすると図 17 に示す学生個別の画面に遷移する. この画面では学習者の早期発見エラーチェックの結果, コンパイルエラー,

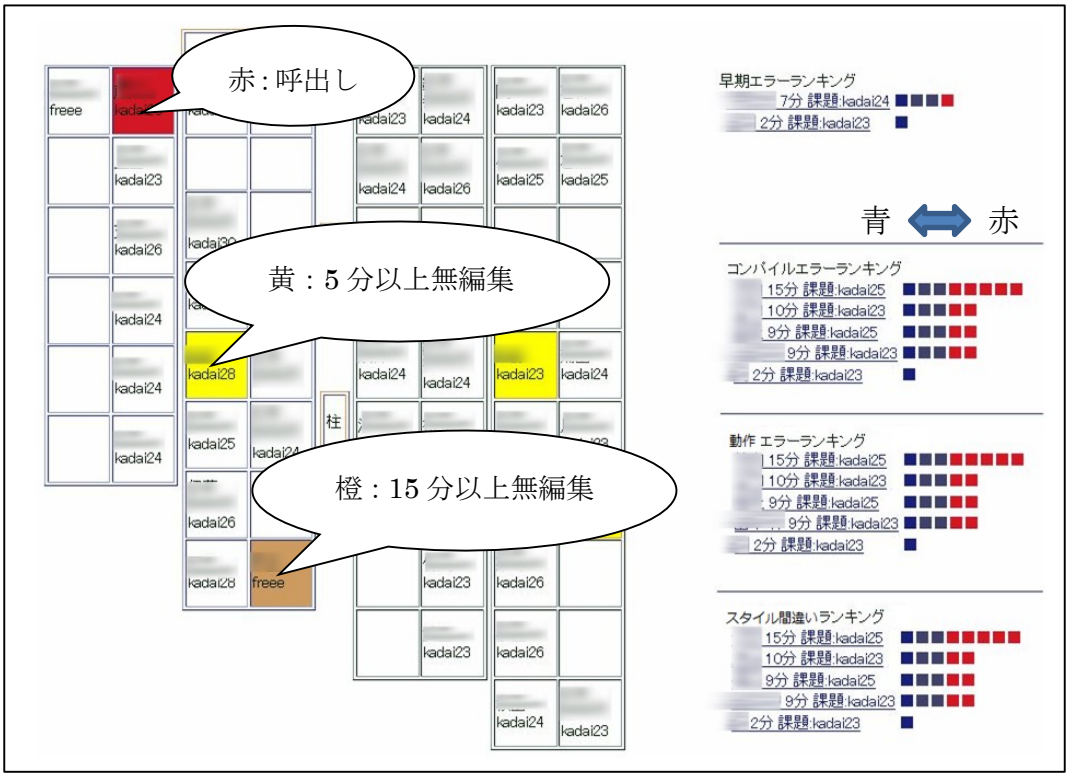

図 16　提案するシステムの画面

プログラムを表示する．講師はこの情報を見て指導に行くべきかどうかを判断する．

例えば，図 17 に示す画面で，学習者のプログラム，コンパイルエラーメッセージを見ると，メソッド名を間違えるという単純なミスをしていることがわかる．しかし画面上部を見ると，9 分間もコンパイルエラーを直せていないことがわかるため，この学習者にアドバイスに行く必要があると判断する．もしエラーを出している時間が短く，エラーの内容も簡単である場合は学習者自身で直せると考え指導に行かないと判断する，

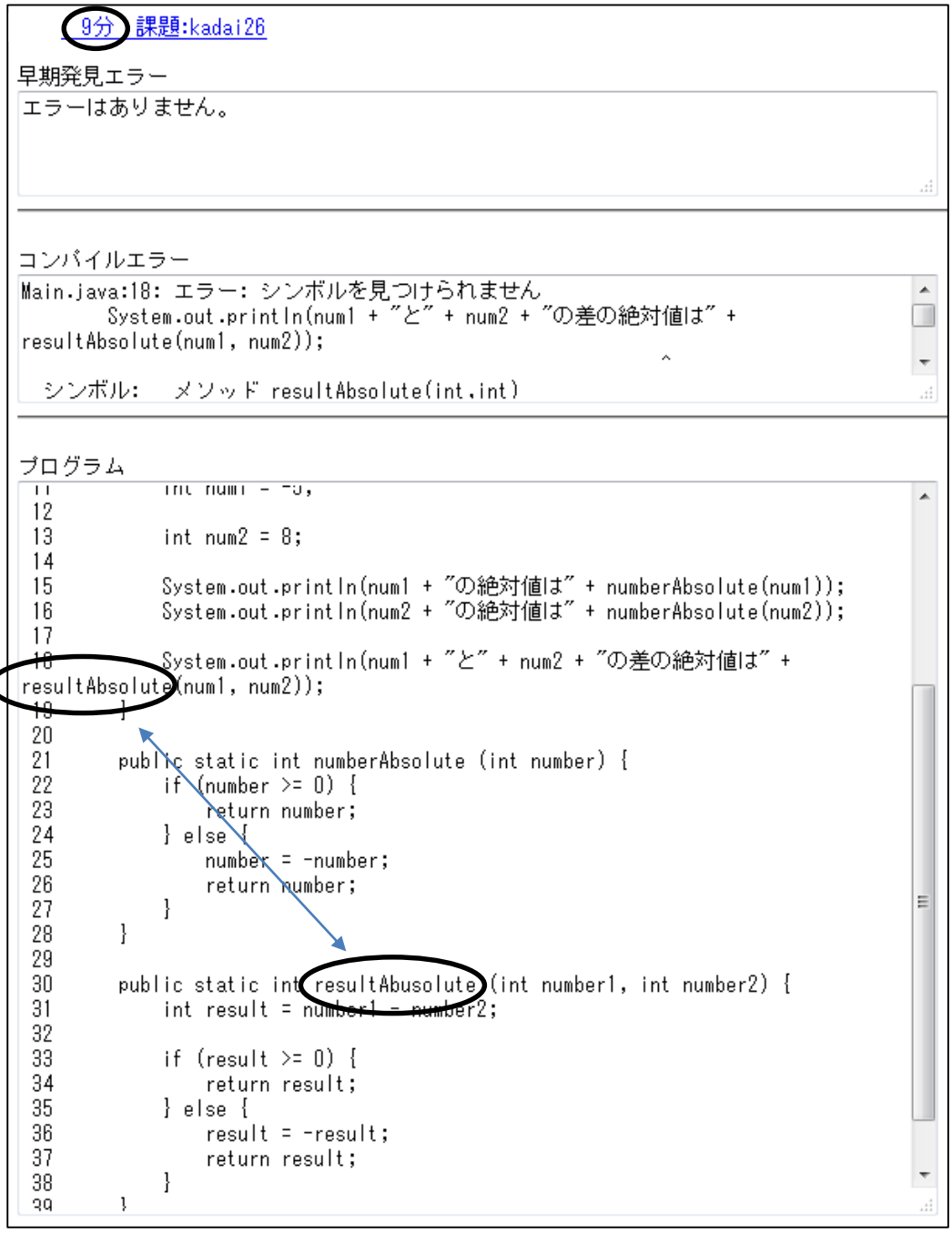

図 17　学習者個別画面（指導に行くべきと判断）

また，例えば，図 18 に示す画面では，全角文字を混入するという間違いをしていることがわかるが，プログラムが作成途中であり，完成してからコンパイルしたときに学習者自身で気づくと考え指導には行かない．

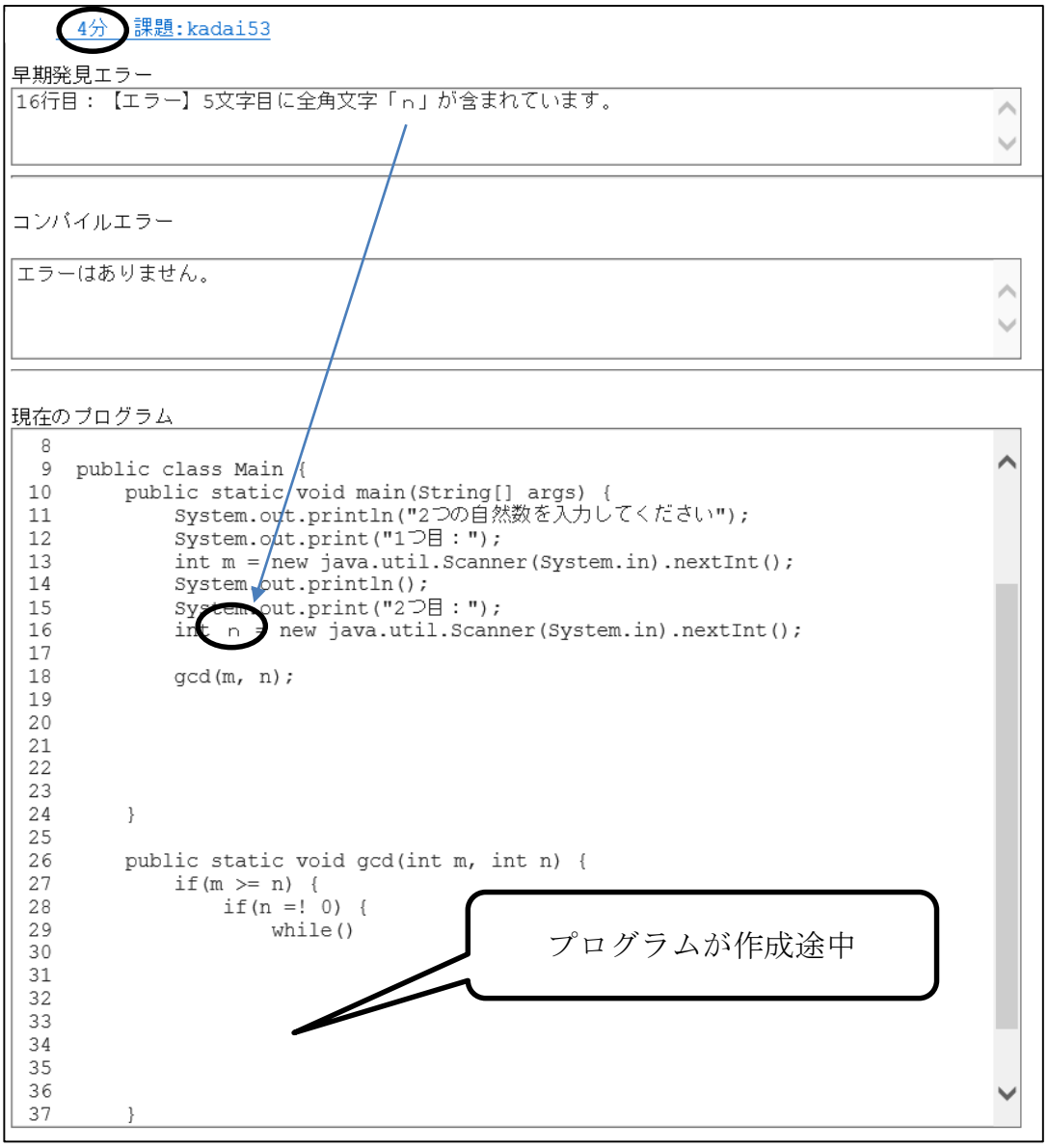

図 18　学習者個別画面（指導に行かなくてもよいと判断）

7 章　運用結果

平成27年度開講の三重大学工学部電気電子工学科2年生を対象の講義であるプログラミング演習 I の 2 クラスにおいて運用を行った．教師には一方のクラスのみ図 16 のように早期エラーチェック，コンパイルエラー一覧を提供し，学習者には両方のクラスで図 3 のように早期発見エラーの結果を提供した，動作エラー，スタイルの間違いについてはこの時点では未実装のため講師には提供していない．同一の演習で，2 つのクラスに対してそれぞれ講師が新システム，旧システムを使用した場合で比較を行った．結果を図 12 と表 3 に，指導した時刻のタイムチャートを図 13 に示す．

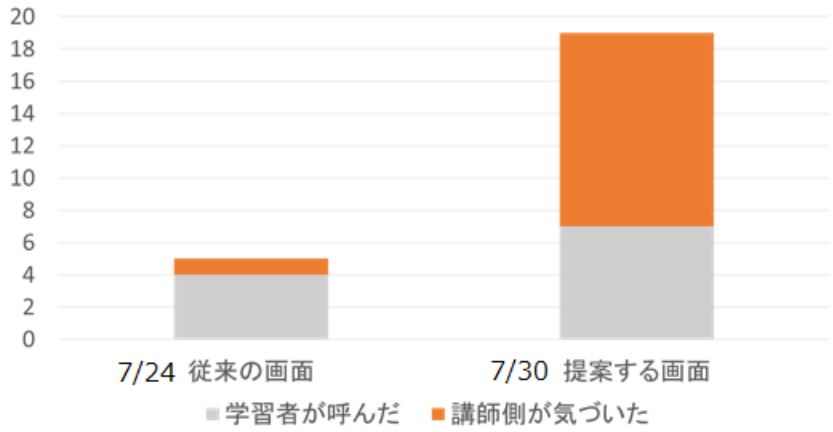

図 19　指導回数の比較

表 3 指導回数の比較.

指導のきっかけ		7/24 従来の座席表 (学習者 44 名)	7/30 提案する座席表 (学習者 44 名)
学習者側	直接呼ばれた	1 回	2 回
	講師呼出しボタン	3 回	5 回
講師側	コンパイルエラー	不使用	11 回
	早期発見エラー	不使用	1 回
	長時間無編集	1 回	0 回

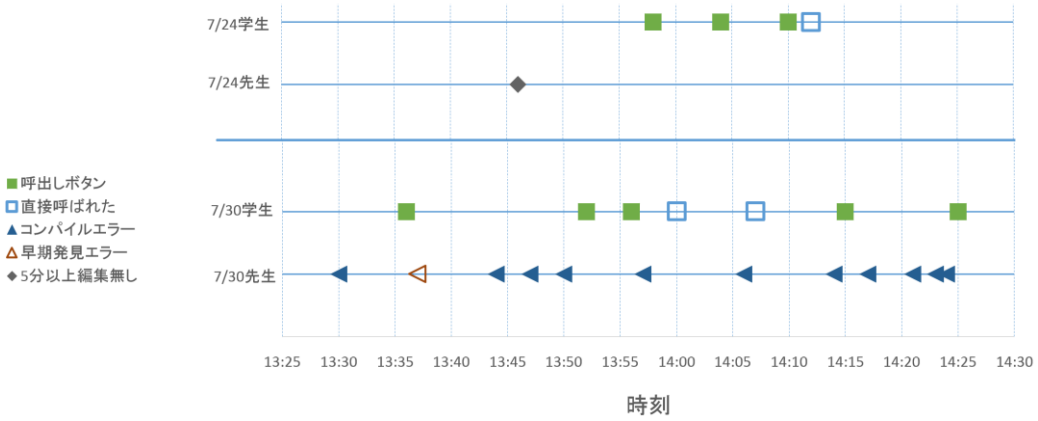

図 20　指導結果のタイムチャート

図 19 の結果から講師側をきっかけとした指導が増えていることがわかり，また，表 3 からは提案するシステムの機能であるコンパイルエラー，早期発見エラーの情報による指導が計 12 回行われていることがわかる．図 20 からは，より早い時刻で指導に行くことができていることがわかる．

また，平成 27 年度開講の三重大学工学部電気電子工学科 2 年生を対象の講義であるプログラミング演習Ⅱのクラスにおいても 2015 年 12 月 17 日に運用を行った．教師には図 16 のように早期エラーチェック，コンパイルエラーランキングを提供し，学習者には両方のクラスで図 3 のように早期発見エラーの結果を提供した，動作エラー，スタイルの間違いについては実装が間に合わなかったため講師には提供していない．結果を表 4 に示す．

表 4 指導回数

指導のきっかけ		12/17 (学習者 78 名)
学習者側	直接呼ばれた	4 回
	講師呼出しボタン	7 回
講師側	コンパイルエラー	8 回
	早期発見エラー	4 回
	長時間無編集	0 回

また，表 4 より同様に，提案するシステムの機能の情報をもとにした指導が計 12 回行われていることがわかる．

提案する機能でも指導できていない学習者がいないかどうか調べるために，提出されたプログラムについて調査をおこなったところ，次のようなプログラムが見られた．

- 冗長なプログラム

  図 21 に同じ課題に対する 2 つのプログラムを示す．右側プログラムのような，一部をコピー&ペーストしたような冗長なプログラムがあった．

```
public class Main {
    public static void main(String[] args) {
        System.out.printf("2 ");
        for (int n = 3; n <= 1000; n++) {
            int i = 0;
            if (n % 100 == 0) {
                System.out.printf("\n");
            }
            for (i = 2; i < n; i++) {
                if (n % i == 0) {
                    break;
                }
            }
            if (i == n) {
                System.out.printf("%d ", i);
            }
        }

    }
}
```

図 21　冗長なプログラムの例

- 変数名が適切でないプログラム

図 22 に学習者のプログラムの一部を示す．正しい英語の綴りとしては「standardDeviation」だが，「standardDeviasion」のように間違えている．

```
int total;
double average;
double workingVariage2 = 0;
double dispersion = 0;
double standardDeviasion;
```

図 22　変数名が適切でないプログラムの例

4 章で述べたように PROPEL には，指導が必要な学習者を表示できず指導できない，指導の時刻が遅いという 2 つの問題点があった．その点が解決できたかについて考察する．

従来指導が必要な学習者を表示できず指導できなかった点に関しては，表 3 の結果より，新システムを使用したときは旧システムでは表示できなかった学習者を表示でき，12 回の指導を行えていることがわかる．また，図 19 より全体の指導回数も増えている．そのため，システムの情報をもとに今まで指導できなかった人に指導できていると考えられる．また，表 4 より，提案する機能が提供する情報であるコンパイルエラー，早期発見エラーの情報がきっかけで指導に行くことができていることがわかる．

指導の時刻が遅かったという問題点について，図 13 より旧システムを使用した場合，指導した時刻が遅い時刻に集中しているが，新システムを使用した場合比較的早い段階で指導できている．よって指導のタイミングが平滑化できていることがわかる．その結果，早い段階で指導を行うことにより，指導が遅い時刻に集中して，指導に十分な時間をかけられないことがなくなることが予測できる．

指導ができなかった学習者のプログラムのうち，図 21 の冗長なプログラムや，図 22 の変数名が適切でないプログラムについては現在提供している機能では検出することができない．どのような情報を抽出すれば良い

かについては検討する必要がある．

　また，システムの不具合により，一覧画面で学習者が正しく表示されない場合があった．これについては今後修正を行う必要がある．

8章　まとめ

プログラミングの授業では，一般に，プログラム作成を行う演習が行われている．しかし，多人数の場合，プログラムの作成が円滑に進まず指導を必要としている学習者を机間巡回で見つけることは困難である．プログラム作成の演習中に，学習者の状況をリアルタイムで把握することができるプログラミング演習システム PROPEL を開発・運用している．このシステムでは講師が iPad 等のタブレット端末上に表示した座席表の画面に表示された情報から，指導が必要と思われる一定時間以上プログラムを変更していない学習者を見つけることができた．

しかし，従来のシステムでは，学習者が一定時間プログラムをまったく変更しなかった場合しか検知することしかできておらず，指導が必要な学習者の一部しか講師はシステムを介して把握することができなかった．その結果，十分な指導を行えていない学習者が存在した．

本研究では，指導が必要な学習者をより多くかつより早く把握できるようにするために，コンパイルエラー，動作エラー，スタイルの間違いを長時間直せない学習者に指導を行う必要があると考えた．本システムではコンパイルエラーまたはコンパイル前に行単位で検出できるエラーが長時間あるいは多数残っている，コンパイル成功後のプログラムが課題の要求に対して正しく動作しない，または，そのスタイルが適切でない学習者のリストを講師に提供した．また，これらの情報はコーディング中あるいはプログラム提出後の学習者にも提供する．

実際の演習でコンパイルエラー，早期発見エラーについて，講師に情報を提供した結果，これらの情報をもとに指導を行うことができることを確認できた．

参考文献

[1] 伊冨昌幸，北英彦，高瀬治彦，林照峯：コーディング状況に応じたアドバイスを可能にするプログラミング演習システムに関する研究，コンピュータ利用教育協議会，2010 PC カンファレンス (2010)

[2] 小川正，西口大亮，北英彦：プログラミング演習における iPad などの携帯デバイスの利用による指導の円滑化，コンピュータ利用教育協議会，2011PC カンファレンス (2011)

[3] 小島佑介，高橋功欣，北英彦：プログラミング演習における効率のよい指導のためのエラー早期指摘，コンピュータ利用教育協議会，2011PC カンファレンス (2011)

[4] Microsoft Visual Studio ホームページ - Visual Studio: https://www.microsoft.com/japan/visualstudio (2015 年 12 月閲覧)

[5] Eclipse - The Eclipse Foundation open source community website：http://www.eclipse.org/ (2016 年 1 月閲覧)

[6] 小川正，西口大亮，北英彦：プログラミング演習における iPad などの携帯デバイスの利用による指導の円滑化，コンピュータ利用教育協議会，2011PC カンファレンス(2011)

[7] 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，55 巻，8 号，pp.1918-1930 (2014)

[8] 倉澤邦美，鈴木恵介，飯島正也，横山節雄，宮寺庸造：プログラミング演習における一斉指導のための学習状況把握支援システムの開発，電子情報通信学会技術研究報告 ET，教育工学 104 (703)，pp.19-24 (2005)

[9] 彦坂知行，北英彦：多人数でのプログラミング演習における学習者のコンパイルエラー状況の可視化，コンピュータ利用教育協議会，2015 PC カンファレンス (2015)

[10] 高橋功欣，小島佑介，北英彦：プログラミング演習における指導のための受講者のコーディング状況の可視化，コンピュータ利用教育協議会，2011 PC カンファレンス (2011)

[11] 柴田 望洋：明解 Java 入門編，SB クリエイティブ (2007)

[12] 北 英彦，望月 将行，高瀬 治彦，林 照峯，森田 直樹：プログラムの自動テスト機能を備えたプログラミング演習システム，日本教育工学会大会講演論文集，20 巻，pp.677-678，(2004)

[13] SourceForge：http://astyle.sourceforge.net/ (2015 年 11 月参照)

[14] 上村拓磨，北英彦：プログラミング演習におけるコード整形ツールを用いたプログラミングスタイルのチェックに関する研究，平成 27 年度三重大学卒業論文 (2016) (予定)

実績一覧

[A] 彦坂知行，北英彦：プログラミング初級者に対するプログラム読解訓練方法，電気関係学会東海支部連合大会，2014/9/9

[B] 彦坂知行，北英彦：多人数でのプログラミング演習における学習者のコンパイルエラー状況の把握システム，PC カンファレンス 2015，2015/8/21

[C] Tomoyuki Hikosaka，Kita Hidehiko: Notification of Existence of Learners who Cannot Fix Programming Errors by Themselves，International Symposium for Sustainability by Engineering at MIU (IS2EMU 2015)，2015/9/24

[D] 彦坂知行，戸上稔崇，上村拓磨，四方雅晴，北英彦：多人数でのプログラミング演習における学習者のコーディング状況の把握システム，CIEC 春季研究会 2016 (予定)，2016/3/26

謝辞

本論文は，著者が三重大学大学院工学研究科博士前期課程に在籍中に行った研究をまとめたものである．本研究を進めるにあたり，懇切丁寧な御指導と御督励を賜った三重大学大学院工学研究科の北英彦准教授に深く感謝いたします．

また，貴重な時間をさいて本論文を査読して頂いた，三重大学大学院工学研究科の森香津夫教授，高瀬治彦准教授に深く感謝致します．最後に，日頃熱心に討論していただいた計算機工学研究室の皆様方にお礼申し上げます．