プログラミング演習における進捗状況の把握及び可視化に関する研究

2013SE059 伊東誠朗　2013SE122 森拓也　2013SE129 長坂総一郎
指導教員：蜂巣吉成

1　はじめに

現在，情報系の大学ではプログラミングの演習が行われている．数十名の学生に対して少数の教員や TA(Teaching Assistant) で対応しなければいけない規模の演習形式が多い．一人一人の学生に対して効率のいい適切な指導を行うことが必要不可欠であり，そのような指導を行うためには学生の進捗状況を教員が詳細に把握することが必要である．一人の学生に割ける時間には限度があるので演習時間内に教員や TA が机間指導で学生すべての進捗状況を把握するのは困難である．類似の指導を個別に複数回行うことは効率が悪く，一斉指導を行うことが望ましい．学生自身が躓いている原因がわからず質問できない場合や質問することに抵抗を感じている場合は気づくのは困難である．

これらの解決策として進捗状況把握システムの利用が挙げられる．教員が机間指導をすることなく学生全体の進捗状況を容易に把握することができ，各学生のソースコード編集過程を見ることができれば，学生全体に効率よく適切な指導を行うことが可能になる．過去の研究では，プログラミング演習時のプロダクトに着目した把握法や，ソースコードを作成する際のコーディング過程やコンパイルといったプロセスに着目した把握方法があったが，作成したプログラムに対して用意された実行例を学生が試すというプロセスに着目したものはなかった．

本研究では，プログラミング演習において，プログラムに対して用意された実行例を学生が試すというプロセスに着目し，学生の進捗状況を把握する方法を提案する．ここでの進捗状況とは，プログラムを作成している学生の課題ごとのソースコードの完成度合いを指す．進捗状況の把握方法としては次の２種類が存在する．本研究の研究対象は前者である．

- プロセスに着目して把握する方法
- プロダクトに着目して把握する方法

ここで言うプロダクトとはプログラミング演習時に学生が作成するソースコードのことである．進捗状況把握方法として学生が実行例を試すというプロセスに着目した研究は過去になかった．実行例が用意されている場合，同値クラスや境界値に基づいていることが多い．様々なソースコードがある中で，学習者の実行例の結果が正しければ同値クラスの処理が正しく，間違っていれば処理が間違っているということが分かる．つまり，教員側は学生のソースコードのおおよその内容が把握できるのではないかと考えられる．

進捗状況把握システムとして次の２点の課題がある．

- 学生全体の進捗把握
- 各学生の進捗把握

学生全体の進捗状況は実行例の正誤による色分け表示で可視化し，各学生の進捗状況はソースコード・入出力・エラーメッセージ等を保存した時間とともに表示することで把握を可能にする．

2　関連研究

井垣らの研究 [1] では，受講生のプログラミング演習時におけるコーディング過程を記録し，可視化して講師に提示するシステムを提案している．課題ごとにソースコードの総行数・コーディング時間・単位時間あたりのエディタ操作数・エラー継続時間の４つのメトリクスを計測し，学生内で順位付けしたものをコーディング過程ビューに可視化することで進捗状況の差を把握している．また，いずれかのメトリクスで相対的に遅れている学生の座席情報に基づいて強調表示することで把握と指導が容易になっている．

安留の研究 [2] では，Web 教材の閲覧履歴や演習課題の採点結果などを学生の席と対応したセルに表示し，タブレット端末で見れるようにすることで，学生個々の進捗状況を確認可能としている．

これらの研究では他の学生と相対的に見て遅れている学生や，模範解答との比較で悩んでいる学生を把握することができる．しかし，躓いている原因の把握を行うことはできない．

高橋らの研究 [3] では伊富らの開発した学生が作成しているプログラムに更新があった最後のタイミングからの経過時間によって色分けし，座席表からそれを確認できるプログラミング演習システム PROPEL の問題点・改良点を挙げ，構文エラー・典型的間違い・動作エラーを受講者のプログラム中の間違いとし，持続時間をタイムチャートによって可視化している．これにより，コーディング手順が可視化され，学生がどのようにプログラムを作成していったかが一目でわかるようになっている．

これらの研究では受講中のソースコードを作成する際のプロセスに着目し，それを可視化することで進捗状況を把握することができる．しかし，本研究の着目点であるプログラムの実行というプロセスからの把握はされていない．

蟹江らの研究 [4] では，進捗状況を学生が書いているソースコードを制御構造，条件式について同値類分割することで学生全体のソースコードの傾向を把握していた．

この研究では [1]，[2] の問題点を解決するための研究が行われているが，ソースコードの傾向の把握を目的としており，可視化を行なっていないので，本研究と合わせることでより詳細な把握が可能になると考える．

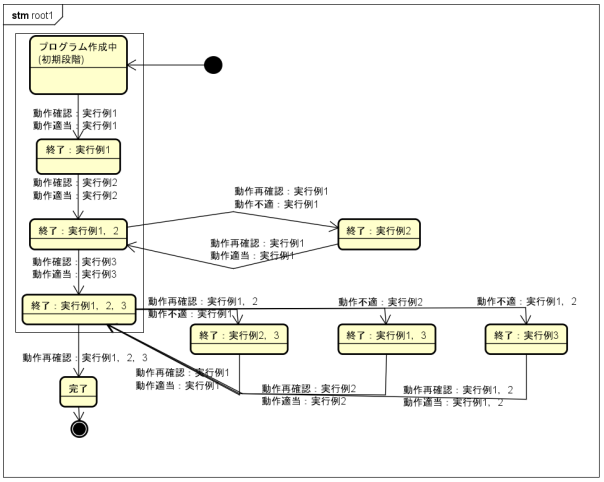

図 1　学生のモデル：プログラム作成の流れ

3　進捗状況把握方法

本研究では，学生が作成したプログラムがあらかじめ用意された実行例について正しく動作しているかどうかを分析し，進捗状況の把握を行う．それによって教員が効率的な指導を行うための補助が可能になる．ここで想定しているプログラミング演習の流れを以下に示す．

1. 教員が学生にプログラムの仕様の書かれた課題と，同値クラスに基づいて定義された実行例を提示する．
2. 学生が，課題のプログラムを作成し，教員に提出する．

本研究では，プログラミング演習において学生は以下のプロセスを経てプログラムを作成すると仮定し，学生がプログラム作成する流れを図 1 のように仮定する．

- 学生が作成したプログラムに対してコンパイルし，エラーが発生した際は修正を行う．
- 作成したプログラムに実行例として用意された値を順番に学生が試し，想定された動作をするか確認する．想定された動作をしなかった場合，修正を行う．

図 1 の状態遷移図では，実行例を試すことによる状態遷移のみを実行例が 3 つの場合で表示している．状態は，どの問題のどの実行例を試したか，どの実行例で正しく動作しているかによって分類される．実行例の数が変更されればその分，状態の数も変化する．コンパイルエラーが発生した場合，コンパイルエラーを修正する状況が発生するが，実行例の正誤から状態を定義しているので，状態としては省略されている．

演習中に想定される学生として，学習者 α，β がいるとする．次の課題が出されているとする．

課題：　実数 x の n 乗 (n は整数) を求める関数 power を作成しなさい．

実行例 1　$x = 2,\ n = 3,\ x^n = 8$

実行例 2　$x = 2,\ n = 0,\ x^n = 1$

実行例 3　$x = 2,\ n = -3,\ x^n = 0.125$

このとき，二人のある時点でのソースコードが図 2，3 であるとする．

```
double power(int x, int n)
{
    double ans=1;
    int i;
    if(n>=0){
        for(i=0;i<=n;i++)
            ans=ans*x;
    }else{
        for(i=0;i>n;i--)
            ans=ans/x;
    }
    return(ans);
}
```

図 2　学習者 α

```
double power(int x, int n)
{
    double ans=1;
    int i;
    if(n>=0){
        for(i=0;i<n;i++)
            ans=ans*x;
    }else{
        for(i=0;i>n;i--)
            ans=ans*x;
    }
    return(ans);
}
```

図 3　学習者 β

学生 α，β が実行例を仮定どおりに試していったとすると，学習者 α は実行例 1 で動作が不適であり，学習者 β は実行例 3 で動作が不適である．このことから，学習者 α は n の値が正の場合の処理で間違えており，学習者 β は n が負の場合の処理で間違えているというように，大まかに間違えている箇所を確認できる．学生が躓いている箇所がある程度把握できることから，学生全体への指導をする際の指針として利用できる．進捗状況の把握が必要な理由として，学生全体のソースコードを教員が逐一確認する時間を短縮する狙いがあるので，これは有効な手段であると言える．

4　進捗状況把握のための学習環境の実現

4.1　要求分析

本研究で取得する必要のあるデータを次に示す．

- 学生の作成しているソースコード
- コンパイル・コンパイルエラーの履歴
- プログラム実行時の入出力

ソースコードを一定時間ごとに保存し，学生のコーディング過程を把握する．ソースコードは 1 分ごとに保存する．

4.2　実現

進捗状況把握のための学習環境を実現するために Web ベースの統合開発環境 WebIDE を利用し，要求分析にしたがって，学生側が使用するページに問題の選択・表示を可能にする機能，ソースコードを 1 分ごとに自動で保存する機能，コンパイル・実行をする機能を実現し，教員側のページでは保存された学生のソースコードから現在と任意の時間のソースコードを表示し比較できる，ソースコード比較機能を実現した．

5　実験

5.1　実験方法

プログラミング演習を想定し，3 年生 11 人に 4 節で述べた環境でプログラムを作成してもらった．その際に取得できたデータから次の 2 点を評価する．

- 学生は実行例を順番に試してプログラムを完成させて

いくという仮定の妥当性

- 学生に対して，どの実行例ができているか分析することで進捗状況の把握ができるという仮定の妥当性

今回，次のような関数を作成する問題を出題した．

問 1　携帯の使用料金を計算，出力する関数 (異なる料金プラン 1，2 が設定されている)

問 2　実数の整数乗を計算する関数

問 3　入力した西暦の年数がうるう年かを判定する関数

問 4　入力された文字列のアルファベットのみを 3 文字ずらす関数

問 1，2，3 は 3 つ，問 4 は 4 つの実行例を用意し，プログラムに対してすべての実行例を学生が試し，全てで正しい動作が確認できた時点でプログラムの完成とした．演習時間は 1 時間としたが学生の状況から 5 分延長した．なお，今回の演習では教員側は演習中の指導は行わないものとした．

5.2　実験結果

実験によって取得できた，各学生のコンパイル，コンパイルエラー，実行時の入出力，問題遷移の時間から分析を行った．学生は課題を問 1，2，3，4 の順番通りに解答していくと思われたが，順番通りに行わない学生が確認できた．

問 2 と問 4 の進捗状況を確認するために，開始から 30 分と 50 分のタイミングに着目し，その時にどの学生がどの問題のどの実行例を試したかを表 1 にまとめる．表内の C エラーはコンパイルエラーを示し，50 分のタイミングの学生 D は用いた Web 環境の不具合で開始から 42 分の時点から継続ができなくなっている．

表 1　30 分と 50 分の状態

	30分		50分	
学生	問題番号	状況	問題番号	状況
A	1	終了	3	編集中
B	4	実1×	1	Cエラー
C	2	編集中	2	実3×
D	2	実3×	4	継続不可
E	2	実3×	4	実1×
F	2	実3×	4	実1×
G	2	終了	4	編集中
H	1	実1, 2, 3×	1	実1, 2, 3×
I	3	実1, 3×	3	実1, 2×
J	2	編集中	2	実1×
K	2	実3×	4	Cエラー

5.3　評価

実験内では用意された実行例以外の数値を試す学生も見られたが，それらすべては用意された実行例と同じ同値クラスの数値であった．被験者への説明時に，「実行例 1，2，3 を試して」という伝え方をしてしまったので，実行例を試す順番が固定されてしまった可能性はあるが，すべての学生が実行例 1，2，3 の順に試し，動作不適なら修正，再び実行といった形で進めていたので，概ね図 1 で仮定した通りであったと言える．

問 1 に関して，学生 G は実行例 1 が上手くいかない状況が開始から 13 分の時点から 5 分程度続いていた．そのソースコードから，実行例 1 にあたる部分の計算式を間違えていることが分かった．

問 2 に関して，30 分の状態では問 2 を解いている学生が 7 人おり，そのうち 4 人の学生 D，E，F，K が実行例 3 の動作が不適の状態にある．この場合，この 4 人は負の整数乗の計算ができていない学生がである可能性が高いので，教師側からは負の整数乗の計算の部分の指導を全体に行うことが望ましいと考えられる．

実際に 30 分の時点での学生 D，E，F，K のソースコードを確認すると，D，E は想定どおりの間違いであり，F，K は int 型等の変数のデータ型の間違いだった．この結果から，想定した指導は概ね適切であったと考えられる．F，K の間違いは実行例から判別しにくいが，ソースコード比較機能によるアプローチにより指導が可能になる．

問 3 に関して，ほとんどの学生がコンパイルが通った後すぐに実行例全てで想定された出力が得られたので，進捗状況把握のための分析はできなかった．

学生 I は開始から 34 分の時点で実行例 2 は正しく動作しているが，実行例 1，3 が上手くいかずに別の問に移っていた．そのソースコードから，閏年を判定する if 文が間違っており，常に同じ出力がされていることが分かった．

問 4 に関して，50 分の状態では問 4 を解いている学生が 5 人おり，そのうち 3 人の学生 E，F，K が実行例 1 の動作不適の状態にある．この場合，文字をずらす処理をで躓いている可能性が高いので，文字列をずらす処理についての指導を全体に行うことが望ましいと考えられる．

実際に 50 分の時点で学生 E，F，K のソースコードを確認すると，3 人とも想定どおりの間違いだった．この結果から，想定どおりの指導が適切であったと言える．

実行例の正誤から個別，または全体への指導の判断を行うことができるので，実行例に着目した分析は有効であると同時に，進捗状況の把握ができると言える．

6　考察

より効率的に学生の進捗状況を把握するためには，可視化についても考察する必要がある．本節では，実験によって得られたデータから考察を行い，おおまかな進捗状況の把握は実行例，詳細を確認するためにソースコード比較ページを用いた可視化方法について提案を行う．

5 章で提示した実験のデータを 1 分ごとに図 4 と図 5 のように学生の状態を表示する．この際，どの実行例を試しどの実行例で正しく動作しているかだけでなく，どの問題のソースコードを編集しているか，他の問題ではどのような状態であるかも表示する．そして，11 人全員の状態が一度に確認できるように表示する．

学生全員の状態を表示した例として，開始から 30 分時点の図 6 を示す．

図 6 のような表示をプログラミング演習中にリアルタイムで行うことができれば，各学生の状態が判別でき，プログラミング演習で指導を行う指針になると考えられる．し

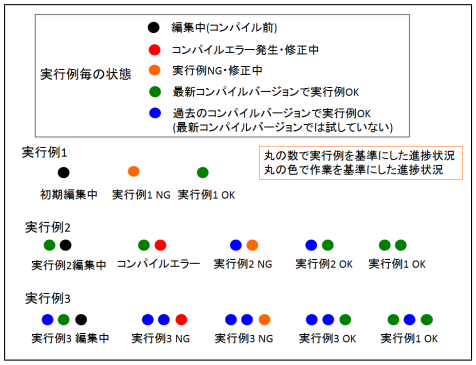

図 4　状態表示 1

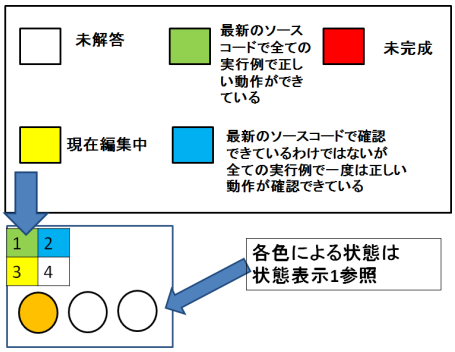

図 5　状態表示 2

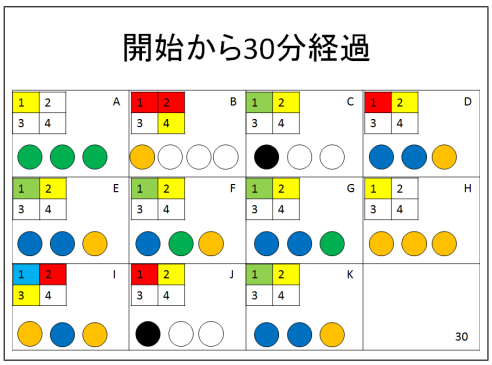

図 6　開始から 30 分の状態

かし，この表示では，ある瞬間における学生全体の状態は分かるが，その問題にかかっている時間などは把握できない．それを補うために図 7 を示す．

図 7 は，現在から 10 分前までの間で学生が行った行動を表示するものであり，1 分ごとに更新される．左右に表示されている数字は，左が 10 分前に解いていた問題の番号で，右は現在解いている問題の番号である．長い時間コンパイルエラーが出ていたり，実行例の動作不適状態が続いている学生は，優先的に指導を行わなければならない対象である．この表示を用いることによって，その判別が可能になる．さらに，ソースコード比較ページにコンパイル履歴，入出力を表示する機能を加える．長い間エラー状態や同じ実行例で動作不適の状態にいる学生に対しては，直接ソースコードを見ることが有効であり，間違えている箇所は履歴からある程度把握できる．全体の進捗を図 6，図

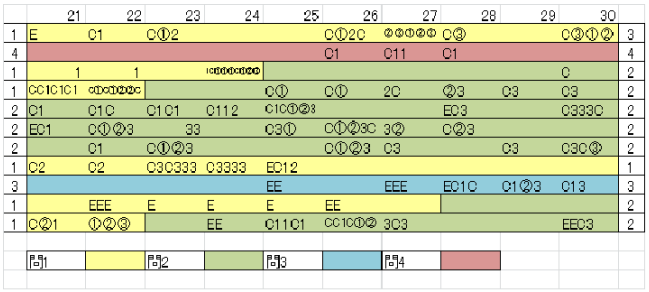

図 7　30 分の時点での流れの表示

7 で把握し，気になった学生を比較ページを用いて確認することで，より効率的かつ的確な指導を行うことが可能であると考えられる．

7　おわりに

本研究では，学生全体及び各学生の進捗状況の把握を実現するために，学生が与えられた実行例を試すというプロセスに着目して可視化する方法を提案した．そのために，学生からリアルタイムにソースコード編集過程を取得し可視化するプログラミング学習環境の実現と，学生が与えられた実行例を試すというプロセスに着目した進捗状況把握が可能かどうかの研究を行った．

実験を通して提案した方法の妥当性を検証した．検証の結果，指導すべき対象や内容の把握を行うことができ，進捗状況の把握が可能であった．今後の課題としては，コンパイル以前のソースコードに対する分析が不十分であることが挙げられる．その解決案として蟹江らの研究 [4] を利用することが考えられるが，どのように行うかは検討が必要である．学生全体の表示をリアルタイムに行う方法ついても今後の課題である．

参考文献

[1] 井垣　宏, 齊藤　俊, 井上亮文, 中村亮太, 楠本真二：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).

[2] 安留誠吾：Web プログラミング演習における学習進捗把握，教育システム情報学会 第 39 回全国大会，H3-3，(2014).

[3] 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，2011PC カンファレンス，(2011).

[4] 蟹江茉衣香，松原知奈美，佐竹玲己衣：プログラミング演習における制御構造と条件式を用いた進捗状況把握方法の提案，南山大学情報理工学部 2015 年度卒業論文，(2016).