プログラミング演習における複数の観点を用いた指導が必要な学習者の特定方法の提案

2016SE034 加藤芳基 2016SE035 加藤祐樹
担当教員：蜂巣吉成

1 はじめに

大学におけるプログラミング演習では，学習者多数名に対して少数の指導者で行われ．指導者が適宜巡回し，質問の対応をしたり，学習者のソースコードを確認して指導が必要だと感じたら指導する．しかし，現在の演習では挙手をしない学習者がいたり，ソースコードの確認に時間かかるので，指導が必要な学習者の適切な発見は困難である．

先行研究 [1][2][4] では学習者のコーディング状況からソースコードの内容，文字数の変化やコンパイルの観点を用いて，何を解答すれば良いか分からず行き詰まる状況，正解と大きく異なるプログラムを編集する状況，コンパイルエラーで行き詰まる状況などの学習者を特定する方法を提案している．しかし，これらの学習者以外にも実行が一致せずに行き詰まる状況，正解と勘違いしている状況，指導者の意図と異なる解答している状況などの学習者に対しても指導が必要だが先行研究では特定できていない．

本研究では，先行研究よりも観点を多く用いることで，より多くの状況の指導が必要な学習者を特定する方法を提案する．観点として，ソースコードの編集，コンパイル，実行に直接関係し観測可能な，模範解答との近づき具合，ソースコードの変化量，コンパイル回数，コンパイルエラー継続時間，実行時エラー判定，入力例の網羅率，テストケースの合格率の 7 つを用いる．これらの観点の組み合わせで，指導が必要な学習者の状況を 8 種類に分類し，それらを発見するための基準を定義した．実際の演習で学習者のコーディング状況を取得して，7 つの観点から 8 種類の状況が特定できるか実験を行なった．

2 関連研究

井垣ら [1] は，オンラインエディタ上の学習者のコーディング過程を分析し，ソースコードの変化量やコンパイルエラー継続時間などの観点を用いて順位付けをし，相対的に進捗が遅れている学習者を把握できる．しかし，ソースコードの内容を見ていないので，学習者がどのような状況で行き詰まっているか把握することは困難である．

藤原ら [2] は，学習者によるコンパイルや実行についての履歴などを計測しコンパイル回数や実行回数，行数などの観点を用いてグラフ化し，指導が必要な学習者の特徴を定量化を行うことでそれに合致した学習者が進捗が遅れていると判断できる．しかし，ソースコードの内容を見ていないので具体的な状況を把握できず，合致した学習者の指導の優先度を決めることは困難である．

久保田ら [4] は，学習者のコーディング状況を進捗度，活動度の 2 つの観点で定義し二次元グラフにすることで，グラフの結果によって指導が必要な学習者を特定している．しかし，この 2 つの観点だと進捗が遅れている学習者や正解まであと少しの学習者は特定できるが，それ以外の状況の特定は困難である．

3 指導が必要な学習者の特定方法の提案

3.1 概略

本研究では，学習者のコーディング状況を把握するために WebIDE を利用する．WebIDE とは，ブラウザ上にてソースコードの記述，コンパイル，実行，作成したソースコードファイルの提出などを行うことができるシステムである．WebIDE で取得した情報の中で学習者のコーディング状況に直接関係する観点を用いる．ただし，問題ごとで得られる観点の数値は，別のものとして扱う．定義した観点の中で先行研究で用いている観点はいくつかある．コンパイルエラー継続時間は井垣ら [1] で用いられている．コンパイル回数は藤原ら [2] で用いられている．模範解答との近づき具合，ソースコードの変化量は久保田ら [4] で用いられているが本研究では，模範解答との近づき具合は正解に対して足りない記述と余分な記述をしていることが分かるように，ソースコードの変化量は長時間の文字数の変化や編集量も分かるように拡張した．入力例の網羅率，テストケースの合格率，実行時エラー判定の 3 つの観点は本研究で新たに用いる観点である．さらに観点を組み合わせて，指導が必要な状況を 8 種類に分けて指導が必要な学習者を特定する方法を提案する．

3.2 観点の定義

3.2.1 模範解答との近づき具合

ある時点でのソースコードが模範解答にどれだけ近づいているかを数値で計測する．方法として石元ら [3] の制御構造の抽象化を用いて変換を行い，制御構造ごとに重み付けを定義し，制御構造の差異で編集距離を取る．これを重み付き編集距離とする．重みは久保田ら [4] で定義されたものを用いる．

模範解答との近づき具合の計算方法は次の通りである．なお本研究では，数値が 0 の場合を制御構造が模範解答と一致していて，正の場合は制御構造が余分にあり，負の場合は制御構造が不足している状況を表す．

$$near(t) = \frac{-x_t}{m}$$

$near(t)$:模範解答との近づき具合

t：問題を解き始めてからの経過時間（$t \geq 1$）

x_t:t 分後のソースコードと模範解答の重み付き編集距離

m:何も書いていない時と模範解答の重み付き編集距離

3.2.2　ソースコードの変化量

WebIDE で取得された 1 分ごとのソースコードから文字数の変化を計測する．久保田ら [4] の 1 分ごとの変化量に加えて，長時間の変化量や編集量を計測する．

ソースコードの変化量の計算は次のとおりである．

$$source(t) = \begin{cases} 0 & (t = 1) \\ a_t - a_{t-1} & (t > 1) \end{cases}$$

$$long\_source(t, n) = a_t - a_{t-n+1} \quad (t \geq n)$$

$$edit\_source(t, n) = \sum_{x=t-n+1}^{t} |source(t)| \quad (t \geq n)$$

t：問題を解き始めてからの経過時間　$(t \geq 1)$

n：時間（t 分経過）

a_t：t 分後のソースコードの文字数

$source(t)$：t 分経過時点での 1 分ごとのソースコードの変化量

$long\_source(t, n)$：t 分経過時点での n 分間のソースコードの変化量

$edit\_source(t, n)$：t 分経過時点での n 分間のソースコードの編集量

3.2.3　コンパイル回数

学習者が行ったコンパイルの回数を計測する．コンパイル回数のうち成功した回数とエラーした回数を計測する．ただし，現在の時刻までで行われた回数とする．

3.2.4　コンパイルエラー継続時間

学習者がコンパイル時にエラーが発生し，以降のコンパイルエラーが発生しなくなるまでの時間を表す．エラーが発生しなくなった場合はエラー継続時間は 0 となる．

3.2.5　入力例の網羅率

各学習者が解答開始時刻から現在時刻までで行った実行結果からテストケースを行なっている割合を把握する（以降は，網羅率と表記する）．網羅率は学習者に提示した実行例のうち，学習者が実行した入力の割合を示すもので，出力が実行例通りかは問わない．

3.2.6　テストケースの合格率

問題で用意されたすべてのテストケースをツール側で実行し，一致したテストケースの割合を計測する（以降は，合格率と表記する）．ツール側で実行する方法として，学習者が行った現在時刻から最も近いコンパイル時のソースコードを用いる．この観点により各学習者のソースコードが正解しているか把握できる．

3.2.7　実行時エラー判定

学習者の現在時刻から最も近い実行結果で実行時エラーが発生したかを判定する．実行時エラーは誤りの箇所が分からないので，行き詰まる可能性が高いので，指導が必要である．この観点により学習者の実行時エラーについての取り組み状況が把握できる．

3.3　指導が必要な状況の定義

状況ごとに観点の数値の基準を仮定し，その基準に該当した学習者は指導対象となる．指導が必要な学習者を状況ごとに定義し，それぞれの状況を特定するために必要な観点の組み合わせを定義する．①，②，③，④は先行研究でも特定できるが，⑤，⑥，⑦，⑧は本研究の新たな観点によって特定できるようになった状況である．指導対象の条件として，1 問あたりの解答の目安時間を半分以上経過した場合とする．目安時間は，過去の類似問題や学習者の習熟度などを考慮して，あらかじめ指導者が設定するものとする．指導対象時間を n 分とする．なお，⑦，⑧は解答時間によって基準が左右されないので，目安時間に関係なく基準に該当する場合は，指導対象となる．

①コンパイルができていない状況

模範解答との近づき具合が低い状況でコンパイルを一度も成功していない学習者は進捗が遅れていると考える．模範解答との近づき具合とコンパイル回数の観点を組み合わせることで特定でき，問題の考え方について指導する必要がある．

基準 1　$near(t) \leq -0.5$ である
基準 2　コンパイル成功回数が 0 である

②何を解答すれば良いか分からない状況

模範解答との近づき具合が上昇せず，ソースコードの変化量や編集量が少ない場合，何を解答すれば良いか分からない状況であると考える．正解との近づき具合とソースコードの変化量と操作量を組み合わせることで特定でき，問題の考え方について指導する必要がある．

基準 1　$near(t) \leq -0.5$ かつ，n 分前と比較して増加していない
基準 2　$long\_source(t, n) \leq \frac{edit\_source(t,n)}{n}$ である

③的はずれな解答をしている状況

ソースコードの変化量が多いにも関わらず，模範解答との近づき具合が上昇しない場合，的はずれな解答をしている状況であると考える．模範解答との近づき具合とソースコードの変化量の観点を組み合わせることで特定でき，問題で問われている意図などについて指導する必要がある．

基準 1　$near(t) \leq 0.5$ かつ，n 分前と比較して増加していない
基準 2　$source(t) > 0$ かつ，$long\_source(t, n) > 0$ である

④コンパイルエラーが解消できない状況

コンパイルエラーが発生してから修正する箇所が分からずソースコードを変化させていない場合やソースコードを修正する箇所以外も削除している場合，エラーが解消できていない状況であると考える．コンパイルエラー継続時間とソースコードの変化量の観点を組み合わせることで特定でき，プログラムの構文などについて指導する必要がある．
基準 1　コンパイルエラー継続時間が 3 分以上である
基準 2　$source(t) \leq 0$ である

⑤実行結果が実行例と一致しない状況

実行結果が実行例と一致せず，ソースコードが変化しない場合や減少している場合，実行が正解と一致せず行き詰っている状況であると考える．合格率とソースコードの変化量の観点を組み合わせることで特定でき，問題のロジックについて指導が必要である．
基準 1　学習者が実行を 1 回以上行っているが，合格率が 100% でない
基準 2　2 回以上連続で $source(t) \leq 0$ である

⑥実行時エラーが解消できない状況

実行時エラーが発生した場合は，エラー内容が表示されないので学習者自身で修正箇所を把握することは困難である．そのためソースコードの変化量が変化しない場合や減少している場合，実行時エラーが解消できない状況であると考える．実行時エラー判定とソースコードの変化量を組み合わせることで特定でき，ポインタの不正アクセスや配列の範囲外参照などについて指導が必要である．
基準 1　学習者の直近の実行結果が実行時エラーである
基準 2　$source(t) \leq 0$ である

⑦正解と勘違いしている状況

テストケースを全て行わず全てのテストケースが一致していない場合に違う問題に移り，その問題に戻らない場合，正解していると勘違いしている状況であると考える．網羅率と合格率の観点を組み合わせることで特定でき，問題を解き直すように指導が必要である．
基準 1　学習者の直近の実行結果から網羅率 < 合格率である
基準 2　違う問題に移動する

⑧別解である状況

学習者のソースコードと模範解答のソースコードが異なるのにも関わらず問題に正解している場合，別解の状況であると考える．模範解答との近づき具合と合格率を組み合わせることで特定でき，解答方法の意図について説明する必要がある．例として必要のない制御構造を記述している場合などがある．しかし，指導が必要でない場合もある．
基準 1　$near(t) \neq 0$ である
基準 2　合格率が 100% である

3.4　指導が必要な学習者を特定するツール

3.3 節で仮定した基準から指導が必要な学習者を特定し，基準に該当する学習者を特定できるか確認するためのツールを作成した．WebIDE で取得されたデータから 3.2 節で定義したそれぞれ観点の計算を行い，任意の時間を指定することで，指定した時間でのそれぞれの学習者の観点の数値，解答時間，解答している問題番号，指定した時間に最も近いソースコードの内容を確認することができる．さらに，指導が必要な学習者を特定するために指導対象になる時間，それぞれの指導が必要な状況の基準を入力することで，任意の時間で基準に該当した学習者の情報に色がつくことで指導が必要な学習者を確認することができる．

ツールで確認できる情報の例を図 1，2 に示す．

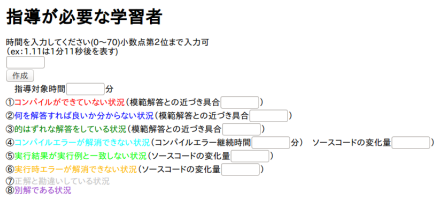

図 1　ツールの初期画面

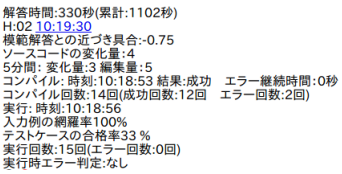

図 2　学習者の情報の例

4　実験

4.1　目的

3 章で提案した方法が実際の演習によりどのくらいの学習者に対して指導が必要になるか調べる．

4.2　方法

学部 3 年生 14 人を対象に行う．問題は全部で 5 問で制限時間は 60 分として，1 問の解答時間の目安を約 10 分とする．

次の関数を作る問題で，1 問あたり 1～3 個の実行例を示した．

問 1　整数の n 乗を計算する関数 (n は非負整数)
問 2　実数の n 乗を計算する関数 (n は整数)
問 3　秒で入力した時間を分と秒に変換する関数

問 4　ユークリッド互除法を用いた再帰関数

問 5　文字列の中に指定した文字が含まれていれば，その文字へのポインタを返す関数

4.3　結果

14 人の学習者データから①〜⑧のそれぞれの状況に該当する回数を調べた．なお，同学習者の同一問題に同じ状況が複数含まれる場合は重複とみなし 1 回とする．本研究の実験では，指導が必要な状況に該当した時点で実際に指導が必要か確かめることができないので，最終的に不正解だった学習者を指導が必要な学習者とする．実験結果を表 1 に示す．

表 1　検証結果

問題番号 / 状況	問 1	問 2	問 3	問 4	問 5	合計	適合率
①	0/0	1/1	1/1	2/2	0/1	4/5	80%
②	0/0	2/2	0/0	1/2	1/1	4/5	80%
③	0/1	1/1	0/0	0/0	1/1	2/3	67%
④	1/1	0/0	1/1	2/2	1/1	5/5	100%
⑤	1/5	3/6	0/1	4/6	1/5	9/23	39%
⑥	0/0	1/1	0/0	1/3	3/7	5/11	45%
⑦	0/0	1/1	0/0	0/0	0/0	1/1	100%
⑧	1/1	1/6	4/4	2/2	0/0	8/13	62%

（不正解者の特定回数 / 特定回数）

全体の結果として，特定された学習者の多くは，状況ごとに想定される指導内容であった．本稿では，新たに特定できるようになった⑤，⑥，⑦，⑧の結果について詳しく記述する．

⑤は，該当者の中で不正解者の多くは，5 分以上連続でこの基準に該当していたので，ソースコードの変化量の基準をより長い時間での変化量についての基準を加えることで，より指導が必要な学習者を特定できると考える．

⑥は，該当者の中で正解に到達した学習者の多くは，実行時エラーが発生してから 3 分程で解消している傾向があった．実行時エラーが発生してから経過した時間についての基準を加え，より指導が必要な学習者を特定できると考える．

⑦は，該当者は解答時間の目安よりもかなり早い約 3 分で解き終わっており，3 つあるテストケースのうち 1 つだけ行い終了し，不正解であった．この結果から，正解と勘違いしている可能性が高いと考える．

⑧は，該当者の中で，8 回は指導者の意図と異なる解答で，指導が必要であると分かった．また，別解ではなかった 5 回は問 2 で検出された．この結果から模範解答の制御構造が複雑な問題の場合，制御構造が少し異なることが多いので，制御構造が複雑な問題に対してはいくつかの模範解答を用意することで別解であるかどうかの判断をより正確にできると考える．

5　考察

指導が必要な学習者の特定方法についての考察を行う．今回の実験により，観点を組み合わせること指導が必要な状況ごとに学習者を特定することができた．特定された学習者のソースコードの内容を見ると，想定された指導内容に該当する場合が多くあった．しかし，特定された学習者の中には，最終的に正解に到達している場合や，同じ時間に複数の学習者が指導が必要な状況に該当する場合がある．本研究で作成したツールでは基準や状況を変更できるので，正解に到達する見込みのある学習者を特定しないように問題ごとにカスタマイズできる．さらに，同時に複数の学習者が指導が必要になる場合は，組み合わせごとの優先順位を決めることが必要で，同じ状況の学習者が多くいた場合は個人指導ではなく集団指導の方が効率よく指導を行うことができると考える．現段階では，指導が必要な学習者を特定した際の指導者への提示方法として，状況ごとに色分けで提示しているが，グラフや表での表示方法など他に見やすい方法を検討する必要がある．

6　おわりに

本研究では，プログラミング演習における指導が必要な学習者の特定方法の提案を行った．コーディング状況を把握するために，模範解答との近づき具合，ソースコードの変化量，コンパイル回数，コンパイルエラー継続時間，合格率，網羅率，実行エラー判定という 7 つの観点を定義して，観点を組み合わせることで指導が必要な学習者の状況を 8 種類に分類して，それらを特定する方法を提案し，提案方法の有効性を実験で確認した．

今後の課題としては，より多くのデータからも検証を行い，観点や組み合わせの妥当性や，より特定できる観点の組み合わせを発見できるか検討する必要がある．

参考文献

[1] 井垣 宏，井上 亮文，齊藤 俊，山田 誠：プログラミング演習における学生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 (2014)．

[2] 藤原理也，田口浩，島田幸廣，高田秀志，島川博光：ストリームデータによる学習者のプログラミング状況把握，電子情報通信学会第 18 回データ工学ワークショップ，pp.1-6 (2007)．

[3] 石元慎太郎, 蜂巣吉成, 吉田敦, 桑原寛明, 阿草清滋：プログラミング演習における構文要素の種類毎のビューによるコーディング状況把握方法の提案，情報処理学会情報教育シンポジウム，2018 論文集，pp.158-165 (2018)．

[4] 久保田 詩門, 蜂巣 吉成, 吉田 敦, 桑原 寛明：プログラミング演習における個別指導のためのコーディング状況把握方法の提案，情報処理学会コンピュータと教育研究会 151 回研究発表会，2019-CE-151(6)，8 pages (2019)．