3ZG-1

ソースコードの履歴を用いた学習者のデバッグ過程の分析

杉山　裕†　　蓑原　隆‡

拓植大学大学院工学研究科電子情報工学科専攻†　　拓植大学工学部情報工学科‡

１．はじめに

情報技術者の教育において，プログラミング技術の学習は必須なものとなっている．プログラミング教育の場において，多くの場合は教育者が学習者の学習状況を把握するために，学習者が作り上げたプログラムソースとプログラム実行結果を提出させる方式が取られている．しかし，この提出された成果物からだけでは学習の手法の傾向や、学習した結果の理解度などの情報を得ることは困難である．そこで本研究ではソースコードの履歴を記録することで学習者の途中経過を得る．しかし取得した全ての履歴を閲覧することは困難である．そこで履歴の中から生徒の情報が把握出来るバージョンの選出を自動的に行う。この選出されたバージョンを閲覧することで、学習者のプログラミング手法や問題解決に対する手法を取得し，学習者の学習傾向を把握することを目的とする．

２．バージョンの重要度と評価値による抽出

本研究では学習者の最終成果物である提出物からでは把握が困難である学習者の学習傾向を把握する為，学習者の途中経過であるソースコードの履歴を利用する．このソースコードの履歴を見ることで学習者がどのような学習経過をたどったのか把握する．しかし，学習経過の情報を得るためには全てのバージョンのソースコードを閲覧するだけではなく，学習者がどのような修正を行ったのかなどを考えながら関連付けていく必要がある．よって，全てのソースコードについて各学習者が抱えた問題の関連付けをする等の作業行うことは非常に困難である．そこで，ソースコードの履歴の中から幾つかの有益なソースコードを見つける手法を提案する．

提案する手法では，履歴の中から学習者の状態情報が把握出来るバージョン(以降チェックポイント)を選出する．選出をする基準として以下の 4 つの事柄が含まれるバージョンは重要度が高い為チェックポイントになると考えた．

1. コンパイル時間間隔が長いバージョン
   学習者が考える時間の長さなので，多くの時間をかけてソースコードを修正した結果は重要度が高い．
2. 変更行数量が多いバージョン
   ソースコードへの多量の変更は問題の解決への手法等の変更，少量の変更は試行錯誤的な修正やテスト用の文面追加等の変更が多い為，変更量が大きいほど重要度が高い．
3. コンパイルエラーが無くなったバージョン
   コンパイルエラーが発生している状態からコンパイルエラーが無くなる状態へ遷移する時点は，学習者がひとまず目指していたプログラムの状態であるから重要度が高い．
4. 最初と最後のバージョン
   学習者の初期状態と，最終到達点はプログラムがどのように変わったのか把握する為に重要度が高い．

これらの理由から，取得した情報を各々の式で数値にして合計することで，評価値を算出する．各評価値を算出する際，1，2については次式で与えるように評価値の増加を非線形にすることで，各項目を過大評価してしまうことを抑制する．

1:$f(x_1) = 1 - exp(-x_1)$　x_1:コンパイル間隔(分)

2:$f(x_2) = 1 - exp\left(-\frac{x_2}{2}\right)$　x_2: 変更行数量(行)

3については評価値 0.5 を与える．ただし，コンパイルエラーが無くなる事が近いバージョンで発生している場合は，減少させた値を与える．4 は評価値 2.0 を与える．この 4 つの評価値の合計が高い順から幾つかのバージョンを選ぶ．

３．評価実験

ソースコードの履歴を取得するために RCS[1] を用いて，本学情報工学科 2 年生の前期のプログラミング実習を受講した 70 名において，学習者がコンパイラを起動する度にソースコードを自動的に記録した．70 名は同一の課題に取り組んでおり課題の内容は，次の 2 つである．

Analysis of Source Code Log for Evaluating Debugging Behavior of Students

†Yutaka SUGIYAMA Electronics and Information Science Course Graduate School of Engineering, Takushoku University

‡Takashi MINOHARA Department of Computer Science, Takushoku University

課題 A： プログラムに対して任意の文字列を 1 つ入力する．この入力された文字列を末尾から先頭に向かうように，逆順に出力するプログラムを作成する．

課題 B： 掛け算九九の問題をランダム生成でするプログラムを作成する．プログラムの終了条件は，ユーザによって入力された数値が 10 回正解するまでとする．

これらの課題を終えた学習者の中から無作為にそれぞれ 5 名，合計 10 名を選び，各々の履歴に提案する手法を用いて評価値を付けチェックポイントを設定した．チェックポイントは優先順位が高いバージョンから各学習者のバージョン総数のうち 20%を四捨五入した整数の個数分を設定した．ただし，初期と最終バージョンはこの数に含めない．図 1 はバージョン毎の評価値の一例であり、横線で示した．閾値以上のバージョンをチェックポイントとして採用する．

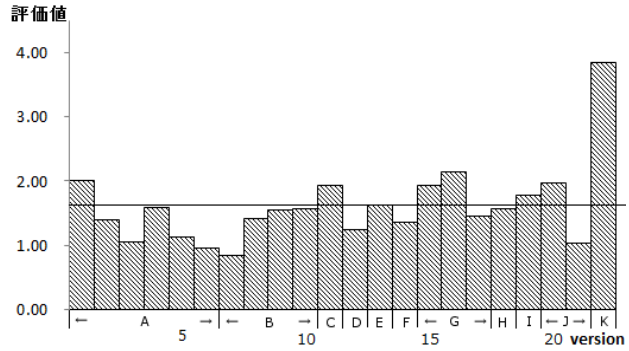

図 1　提案手法による学習者 A の解析結果

提案する手法によって選択されたバージョンが適切かどうかを評価するため全バージョンのソースファイルを確認し，学習者の作業内容が変化したと考えられる点でバージョンを分割し，分割された区分を選択したバージョンがカバーしているかどうかを評価する．手作業による分割の結果例を図 1 の横軸下段のアルファベットにて示した．この手作業で分割した各区間内で 1 つのバージョンを見れば，学習者のつまづいた原因が把握出来ると考える．このときの手作業による分割区間数，提案手法で設定されるチェックポイント数，有効なチェックポイント数，有効率を課題 A は表 1，課題 B は表 2 に示す．ここで有効なチェックポイント数とは，手作業による分割された区間内で，提案手法によって設定されるチェックポイント数は 1 つまで有効とした数である．有効率は，提案手法によって設定されたチェックポイントのうち，どれだけ有効であったかの割合を示す．なお，表中において，CP はチェックポイントのことである．

表 1　課題 A の結果

学習者 (N)	手作業による区分数	提案手法による CP 数	有効な CP 数	有効率
A	11	6	6	100.0%
B	10	8	7	87.5%
C	12	8	5	62.5%
D	12	11	9	81.8%
E	8	15	7	46.6%

表 2 課題 B の結果

学習者 (N)	手作業による区分数	提案手法による CP 数	有効な CP 数	有効率
F	13	12	8	66.6%
G	37	15	14	93.3%
H	18	11	9	81.8%
I	27	12	11	91.6%
J	11	7	5	71.4%

実験を行った結果，学習者は比較的多くの作業内容が変化することがわかった．そのため，提案する手法におけるチェックポイント数は手作業で分割した区分数より少ない．しかし有効率については平均 78.31%である．よって指定個数分だけでは生徒の学習傾向の把握が困難になる場合は，次に評価値が高いポイントを追加参照すれば補完することができると考える．

4．まとめ

学習者の学習傾向の把握をする為に，学習者がコンパイルを行うたびにソースコードの履歴を取り，履歴利用することを述べた．この履歴を全てのバージョンを見ることは困難であることを述べた．この問題を解決するために，ソースコードの履歴の中から優先度の高いソースコードを抽出する，評価値によるチェックポイント設定方法を述べた．この手法によって実験を行い，手作業による解析結果と比較し 78%以上チェックポイントが有効に設定できる事を示した．よって，チェックポイントのソースコードを見れば，学習者がつまづいた場面や解決した手法を高い確率で把握することができる．よって本研究の目的である教育者が学習者の傾向などの情報を取得することができる．

参考文献

[1]Walter F.Tichy:“RCS—A System for Version Control.” Software—Practice and Experience. Volume 15, Number 7, pp.637-654 (1985)