Vol.2013-CE-118 No.8
2013/2/8

コンパイルエラー修正時間に着目した学習分析指標の提案と内省学習効果分析への適用

榊原　康友[1,a)]　松澤　芳昭[2,b)]　酒井　三四郎[2,c)]

概要：プログラミング学習において内省が有効であるとする先行研究がある．しかしながら，先行研究では学習者の内省記述のみから学習効果を分析しており，コンパイルエラー修正時の行動を量的に測れないことが問題である．そこで，本研究では修正時間に着目したコンパイルエラーの難易度及び理解度を分析する指標を提案する．この指標をプログラミング初学者が課題を解く過程で得られた学習記録へ適用することで内省学習効果の分析を試みた．1) 失敗知識の内容から理解度の向上が窺える時は，その次の修正時から修正時間が短くなる傾向を持つことが多い．2) 非内省学習者と比較して内省学習者の修正時間は短い時間で収束しやすい．3) 非内省学習者と比較して内省学習者は短い時間で収束する速度が速い．

1. はじめに

C や Java のようなテキストベースのプログラミング言語の導入学習において，コンパイルエラーの問題は避けられないものである．プログラミング初学者には，構文などの言語仕様を理解していないためコンパイルエラーが発生し，修正で躓く人もいる．そこで，テキストベースのプログラミング言語の学習においてはコンパイルエラーの修正方法を学習させる必要がある．

コンパイルエラーの修正方法は，エラーに直面しそれに対処する過程で理解が深まることが多い．しかしながら，失敗をその場限りのものとしてしまい，次に活かすことができない問題がある．先行研究 [1] ではこの問題に有効であるのが内省学習だと主張しており，筆者らもそれに賛同している．更に，内省学習を行うことで知識が定着しやすくなり，エラー修正方法の理解が促進されやすく，内省学習を行わなかった時よりも短期間でエラーの修正が容易になると考えられる．しかし，先行研究では学習者の内省記述のみから学習効果を分析しており，量的に評価できていないことが問題である．

コンパイルエラーの修正難易度指標はエラー数ではなく修正時間を重視するべきである．エラーが発生しても，すぐに修正できれば修正難易度は低いと考えられるため，エラー数を指標とするのは不適切だと考えられる．一方，修正時間を指標にすることで修正時間が長ければ学習者が修正するのに困難なエラー，修正時間が短ければ修正するのに容易なエラーだと仮定できる．

本研究では，プログラミング初学者を対象に内省学習を利用したコンパイルエラーの学習支援を行う．更に，コンパイルエラーの修正時間に着目したコンパイルエラー分析指標の提案を行い内省学習効果分析に適用した．本論文では，内省支援システムの概要，内省学習効果の分析結果について述べる．

2. 先行研究

2.1 コンパイルエラー分析

コンパイルエラーの修正難易度を示す指標を提案している研究がある．

Matthew らは，プログラミング学習者が発生させた Java 言語のコンパイルエラーの分析を行っている [2]．この研究では学習者が構文解析エラーの修正に対してどれだけ苦労したのかの指標を示す EQ(error quotient) を定義している．EQ はコンパイルした時に連続してコンパイルエラーが発生したか，連続して発生したコンパイルエラーが同じエラーメッセージであるかを利用して算出している．

EQ はエラーの修正難易度を示す指標としては不十分である．何度も連続してエラーが発生しても短い時間で修正できれば修正の難易度はそれほど高くはないと考えられ，連続してエラーが発生しなくてもエラーを修正するのに長い時間が必要だったのならば修正の難易度は高いと考えら

1 静岡大学大学院情報学研究科
Graduate Schools of Informatics , Shizuoka University
2 静岡大学情報学部
Faculty of Informatics , Shizuoka University
a) gs11021@s.inf.shizuoka.ac.jp
b) matsuzawa@inf.shizuoka.ac.jp
c) sakai@inf.shizuoka.ac.jp

れる．

コンパイルエラーに対する学生の反応を評価する基準を提案した研究がある．

Marceau らは，プログラミング初学者のコンパイルエラーのメッセージに対する反応の分析を行っている [3]．学習者の反応を評価する基準を定義し，9 種類のコンパイルエラーについてエラー修正の難易度を分析している．

この基準の定義からエラーの修正難易度を算出するのは不十分である．学生がエラーに対してすぐにその反応を示したのか，時間が経ってから反応を示したかにもよってエラーの修正難易度は異なってくると考えられる．

プログラミング学習者が発生させたコンパイルエラーの内容を手動で分析した研究がある．

Jackson らは，Java 言語のコンパイルエラーを自動的に収集するシステムの開発を行っている [4]．システムによって集められたコンパイルエラー数の分析と，コンパイルエラーにおける指導者と学習者の相違の分析を行っている．

梶浦らは，プログラミング学習者が発生させた C 言語の文法エラーの分析を行っている [5]．この研究では文法エラーメッセージの数を集計するだけでは学習者が引き起こした文法エラーの内容は分からないとし，学習者 50 名のソースコードから実際のエラー内容を分析している．

森本らは，プログラミング学習支援データを作る目的で C 言語のコンパイルエラー内容の分析を行っている [6]．この研究ではそれぞれのコンパイルエラーメッセージに対して学籍番号やコンパイルエラー時間などの基本情報に加え，エラー要因や関連エラーといった情報を手作業で付加したものを学習支援データとしている．

これらの研究では学習者のログを全て手作業で分析しているため効率が悪い．同じコンパイルエラーのメッセージであっても異なる原因で発生する場合もあるため，手作業でエラーを分析することは有効である．しかしながら，学習者が発生させるエラー数は膨大であるため，それらすべてに対して手作業で分析することは無駄である．そこで，学習者にとって修正が難しかったエラーが予め分かれば分析の効率が上がると考えられる．

2.2 エラー学習支援

エラー修正の理解促進の支援を行う研究がある．

知見らは，内省を利用したプログラミングエラーの支援を行っている [1]．プログラミング学習には内省が効果的であるとし，学習者がエラーを発生させた時にエラーの推定原因，確定原因，対処，総括を記述させることによって内省の支援を行っている．

この研究では学習者の言語仕様の理解を内省の記述から質的な評価をしている．しかしながら，量的な分析をしていないため，実際に学習者がエラーに対してすぐに修正できるようになったのか分からない．

Hristova らは，Java 導入教育用のプログラミング開発環境の開発を行っている [7]．コンパイラが表示するエラーメッセージを初学者にも分かりやすいものに変更することで支援している．プログラミング初学者が躓きやすいエラーのリストを講義の様子から作成し，そのリストに含まれるエラーに対応するエラーメッセージを変更している．

この研究ではエラーのリストを講義の様子から作成しており，指導者の主観で躓きやすいエラーを選出している．しかしながら，選出されたエラーが本当に学習者にとって躓きやすいエラーであったか分からない．そこで，エラーの収集は主観ではなく客観的に示された指標から行うべきである．

3. 分析指標の提案

3.1 修正時間を利用した指標

修正時間に注目したコンパイルエラーの難易度分析指標を提案した．本稿では説明しないため「プログラミング初学者におけるコンパイルエラー修正時間とその増減速度の分析」[8] を参照すること．

3.2 修正時間の変化タイプ

学習者における修正時間の変化は以下の 4 つのタイプに分類できる．

3.2.1 Type1：収束

Type1 はコンパイルエラーの修正回数が増えるにつれて修正時間が短くなっていき，最終的に修正時間が短時間で収束するタイプである．修正時間が短時間で収束しているかの基準は修正時間の最小値に 20 秒を加えた値以下の修正時間で連続で 5 回以上修正していることと定義する．Type1 の例を**図 1** に示す．図 1 では，エラー修正数が 15 回目までは修正時間が長いものが多かったが，エラー修正数が 16 回目以降に 8 回連続でほぼ一定の短い時間で修正しており，収束基準を満たしている．そのため，図 1 では 16 回目から収束する傾向があると言える．

3.2.2 Type2：未収束

Type2 はコンパイルエラーの修正数が増えても，修正時間が短時間で収束しないタイプである．Type2 の例を**図 2** に示す．図 2 では，修正時間が短い時もあるが，短時間で収束している傾向が見られないため，未収束のタイプとなる．

3.2.3 Type3：短時間

Type3 はコンパイルエラーを初めて修正した時から，修正をこなしても終始修正時間が短いタイプである．Type3 の例を**図 3** に示す．図 3 のように，初めてのコンパイルエラー修正にもかかわらず修正時間が短く，これ以降の修正でも主に短時間で修正をしているものがこのタイプになる．

3.2.4 Type4：傾向不明

Type4 はコンパイルエラーの修正数が少なく，修正時間

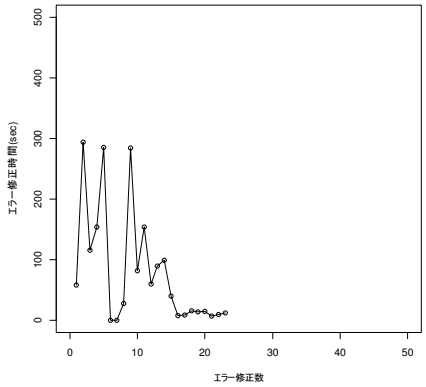


図 1 Type1 の例

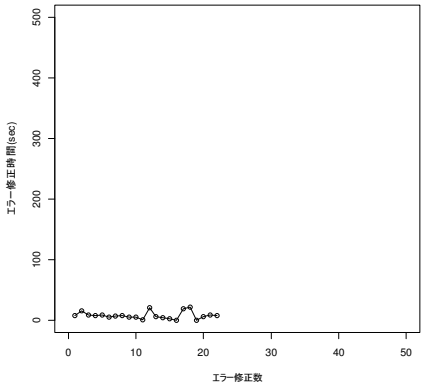


図 3 Type3 の例

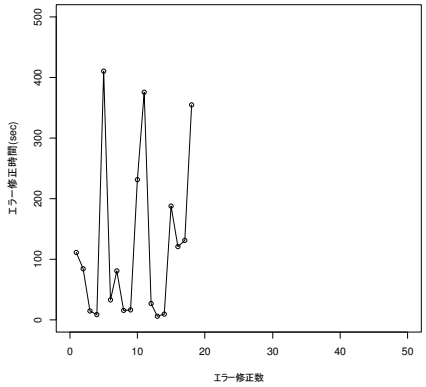


図 2 Type2 の例

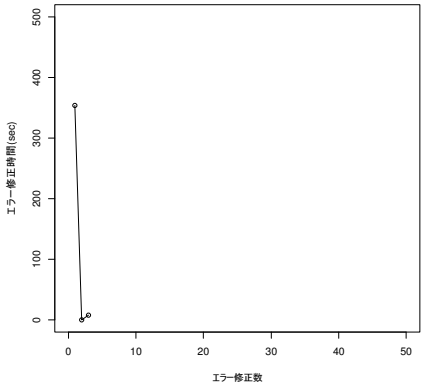


図 4 Type4 の例

も長かったり短かったりしているため傾向を把握できないタイプである．Type4 の例を**図 4** に示す．図 4 のように修正数が少なく，修正時間が長い時と短い時もあるため，これ以降に収束するか不明のものがこのタイプである．

4. 内省支援システム：GeneRef

4.1 GeneRef

GeneRef は，コンパイルエラーの修正方法についての内省を促し，内省を支援するためのアプリケーションソフトウェアである．本ソフトウェアの目的は，学習者にコンパイルエラー修正方法について内省を促すことで，学習者が短期間でコンパイルエラーの修正方法の知識を定着できるようになること，更に，内省しやすい環境を提供することである．

内省は，失敗をその場限りのものとしてしまうことで問題の本質を理解しないまま学習を終えてしまい，次に活かすことができないという問題に対して有効であると考えられる．それに加え，内省をすることによって知識の定着が早くなると考えられる．コンパイルエラー修正方法の知識が定着すれば，短時間でエラーを修正できるようになると考えられる．

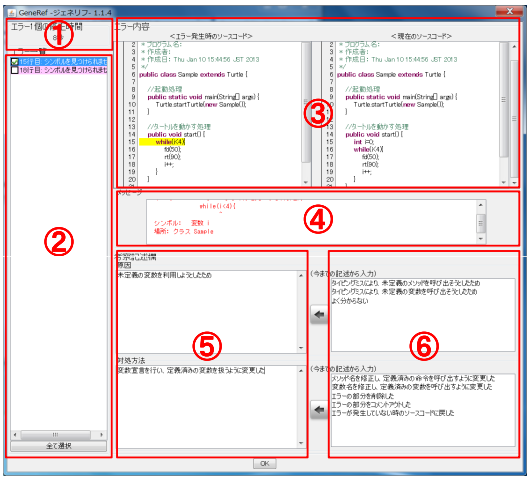


図 5 GeneRef スクリーンショット

4.2 システムの概要

GeneRef は学習者が発生させたコンパイルエラーが修正された時，修正されたエラーについての「原因」と「対処方法」（以下，原因と対処方法のことを併せて失敗知識と呼ぶ）を記述する機会を与えることで内省を促すシステムである．GeneRef のユーザインタフェースを**図 5** に示す．図 5 内の，数字を記入した場所の名称と機能を以下に列挙する．

(1) 修正時間ビュー．エラー 1 個あたりの修正にかかった時間が表示される．
(2) 修正エラーリスト．修正されたエラー一覧が表示されており，ここからエラーを選択した状態で失敗知識記述欄に失敗知識を書き込むことで，選択したエラーについての内省記述が完了する．記述が完了したエラーは左側のチェックボックスにチェックが入る．また，エラーは複数を選択することが可能であり，複数選択した状態で書き込むと選択したエラー全てに同一の内容が記述される．
(3) ソースコードビュー．コンパイルエラー発生時のソースコード（左）とコンパイルエラー修正時のソースコード（右）が表示される．
(4) メッセージビュー．選択されているエラーのエラーメッセージ（コンパイラが出力したメッセージそのもの）が表示される．
(5) 失敗知識記述欄．失敗知識を書き込むテキストエリアがある．
(6) 失敗知識リスト．過去に記述した失敗知識を選択することができ，選択すると失敗知識記述欄に同一内容の記述が書き込まれる．

GeneRef は学習者が利用するプログラミング用のエディタに実装しており，コンパイルエラーが修正されると図 5 の画面が表示される．また，修正時間のスレッショルドを設定することで，コンパイルエラーの修正時間がスレッショルドの値よりも長かった時にだけ画面を表示させるようにすることもできる．エラーの修正方法を理解しているが，うっかり意図したことと異なる行動をしてしまったことが原因でエラーを発生させてしまうことはよくある．そのような場合に内省を促したとしても，すでに修正方法は理解しているため内省効果が望めないだけでなく，学習者に内省の記述時間を取らせてしまうため煩わしさのみが生じてしまう．そこで，スレッショルドを設定することでそのようなエラーは内省を行わなくてもよい仕様とした．スレッショルドを設定する際，個々の学生によって能力が異なるため修正時間ビューを参考に各自でスレッショルドを調整できるようにした．

5. 実験概要

大学生のプログラミング教育現場で GeneRef を導入し，得られたエディタの操作履歴データに提案指標を適用することで内省学習効果の分析を行った．本章ではその概要を述べる．

5.1 対象

対象は Java プログラミング入門の講義を受講している文科系の大学 1 年生約 100 名であり，対象者全員からエディタの操作履歴データを収集した．これらのデータは講義 11 回分で課された課題を解く過程で得られたものである．

表 1 各課題における学習目標

番号	学習目標
1	タートルのメソッド使用
2	変数，条件分岐，入出力，乱数
3	繰り返し，入れ子構造，余り演算子
4	オブジェクト
5	画像表示，キーボード操作
6	実数型
7	なし（自由課題）
8	キャスト
9	メソッド作成，引数
10	戻り値
11	再帰関数

GeneRef の導入は実験当初，プログラミング入門の講義を受講している学習者からランダムサンプリングで選んだ半数の学習者だけに利用する予定であった．しかし，失敗知識を見ると適切な使い方をしていた学習者はあまり多くなく，ほとんどの学習者が GeneRef を利用しないと同じ状況になっていた．

5.2 プログラミング環境

対象者全員がエディタ，コンパイラを統一してプログラミングを行っている．コンパイラは Oracle 社が提供する Java Development Kit に含まれる Javac コンパイラで，バージョンは 1.7.0_07 である．更に，本講義ではビジュアルプログラミング言語を取り入れており，それを利用して課題を解く者もいる．

5.3 課題内容

課題 6, 課題 8, 課題 10 以外の課題は講義用に機能が追加されたタートルライブラリを利用した課題となっている．各課題には学習目標が設定されている．**表 1** に各課題における学習目標を示す．課題にはテキストエディタを利用して解く課題，ビジュアルプログラミング言語を利用して解く課題，どちらを利用して解いてもよい課題がある．

6. 結果

6.1 データ分析概要

6.1.1 コンパイルエラー分析概要

コンパイルエラー総数は 77,885 個，エラーの種類は 79 種類であった．エラー 1 個の平均修正時間は 29.4 秒であった．

6.1.2 失敗知識分析概要

失敗知識の登録数は 9,762 件，失敗知識の記述機会は 4,519 回であり，エラー 69 種類に対して失敗知識が記述された．GeneRef を 1 度でも利用して失敗知識を記述した学習者は 52 人であった．ただし，失敗知識の記述はいい加減に書く人もいるため，必ずしも全ての GeneRef 利用者が GeneRef で内省を行っているとは限らない．以下，内省

表 2 理解度と修正時間の減少有無

エラーメッセージ	修正時間減少 ○	修正時間減少 ×
クラス A のメソッド B は指定された型に適用できません	4	1
構文解析中にファイルの終わりに移りました	10	3

学習者とは GeneRef を利用して内省を行った学習者を指し，非内省学習者とは GeneRef を利用していない学習者と GeneRef は利用したが，その記述内容から内省を行っていないと筆者が判断した学習者のことを指す．

6.2 内省学習の分析

内省学習の効果を失敗知識の記述内容とコンパイルエラー修正時間から分析した．「構文解析中にファイルの終わりに移りました」「class,interface または enum がありません」の 2 種類のコンパイルエラーにおける修正時間の変化を表わしたデータそれぞれ 20 件を付録の**図 8** と**図 9** に付す．これらの図の縦軸は修正時間，横軸はエラー数を表わす．また，図下部には学習者識別番号，内省学習者を表わす G または非内省学習者を表わす NG，修正時間の変化タイプを記した．

6.2.1 修正時間と理解度の関連

学習者のコンパイルエラー修正方法の理解度と修正時間の関連について分析を行った．エラーの修正方法が分からないものでも，修正経験を重ねることによって修正方法が分かるようになる．失敗知識の記述において，最初は「分からない」と記述していたが，修正経験を重ねることで適切な記述ができるようになり，理解が深まる様子が見て取れるケースがある．理解度と修正時間の減少関係を**表 2** に示す．表 2 中の○は理解が深まったと同時に修正時間が短くなるケース数を，×は理解が深まったけれども修正時間が短くならないケース数を示している．

エラーメッセージ「クラス A のメソッド B は指定された型に適用できません」における，失敗知識の記述から窺える理解度が向上するケースは 6 例あった．そのうち 1 例は，「分からない」と記述しているものの，Type3 の特徴を持っていた．6 例中 4 例は「分からない」と記述していた期間は修正時間が長くなる傾向が見られたが，適切な記述をし始めたエラー修正を境に，それ以降の修正時間が短くなる傾向が見られたケースである．残りの 1 例は適切な記述になっても，修正時間が短くならないケースである．

エラーメッセージ「構文解析中にファイルの終わりに移りました」における，失敗知識の記述から窺える理解度が向上するケースは 15 例あった．そのうちの 2 例は，Type3 の特徴を持つケース，10 例が修正時間が短くなる傾向のあるケース，3 例が修正時間が短くならないケースであった．

図 6 では，エラー修正数 1 回目から適切な記述をしている．これによって，8 回目までは修正時間が減少しているが，9 回目のエラー修正時間が増加している．これは，9 回目の失敗知識が「分からない」になっていることから，修正方法を忘れてしまったためだと考えられる．その後，12 回目の修正（2 個目の赤色プロット）で再び修正方法を思い出し，適切な記述に戻ると同時に修正時間が短くなっている．

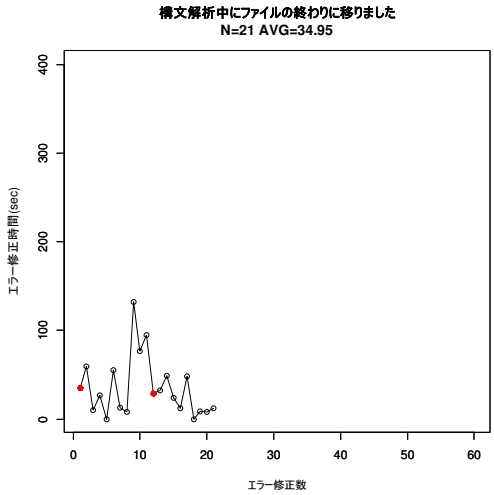


図 6 修正時間減少例 2

6.2.2 Type1 と Type2 の数の比較

内省学習者と非内省学習者の Type1 と Type2 の数を比較を行った．比較対象のコンパイルエラーは Type1 の特徴を多く持つ「class,interface または enum がありません」「構文解析中にファイルの終わりに移りました」の 2 種類のメッセージである．Type 比較において，Type3 は最初から修正時間が短いため，内省効果とは無関係だと考えられるため，Type4 はエラー数が少なく特徴を捉えられず比較できないため除外した．結果を**表 3** に示す．

表 3 より，Type1 の割合はエラーメッセージ「class,interface または enum がありません」においては内省学習者が約 82%，非内省学習者が約 70%，エラーメッセージ「構文解析中にファイルの終わりに移りました」においては内省学習者が約 91%，非内省学習者が約 80%と，どちらのコンパイルエラーにおいても，内省学習者の Type1 の割合は非内省学習者の割合よりも高いことが分かった．

6.2.3 修正時間収束までのエラー修正数の比較

Type1 の特徴を多く持つエラーメッセージ「class,interface または enum がありません」において，内省学習者と非内省学習者で，修正時間が収束するまでのエラー数を比較した．その結果を**図 7** に示す．

表 3 Type1 と Type2 の数比較

	class,interface または enum がありません		構文解析中にファイルの終わりに移りました	
内省学習	Type1	Type2	Type1	Type2
○	9(82%)	2(18%)	20(91%)	2(9%)
×	46(70%)	20(30%)	58(80%)	15(20%)

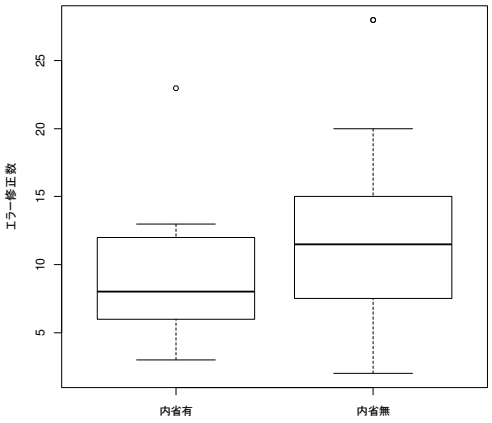


図 7 修正時間が収束するまでのエラー修正数

図 7 より，内省学習者の方がエラー修正数が少ないことが分かる．これは，内省学習者の方が非内省学習者よりも修正時間が収束しやすいことを示す．

7. 考察

分析結果より，以下の 3 つのことが分かった．1) 失敗知識の内容から理解度の向上が窺える時は，その次の修正時から修正時間が短くなる傾向を持つことが多い．2) 非内省学習者と比較して内省学習者の修正時間は Type1 になりやすい．3) 非内省学習者と比較して内省学習者は Type1 の収束する速度が早い．

しかし，これらの成果が全て GeneRef の効果であると言いきることはできない．GeneRef の特徴上，コンパイルエラーが修正された時，一時的に学習者のプログラミングの作業を中断させることになる．これによって，記述が煩わしく感じられる学習者も多くいたが，内省学習者は記述を適切に行っている．このことから内省学習者はプログラミングが得意な，あるいはプログラミングに取り組む意識が高い学習者であり，失敗知識の記述を適切に行った学習者は優秀である者が集まったとも考えられる．プログラミングは個人差での影響が大きく関係してくると考えられるため，必ずしも GeneRef の効果によって結果が表れたと言えないのである．

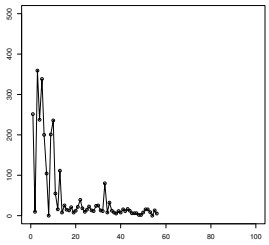
しかしながら，本研究では内省学習にコンパイルエラーの修正時間に着目した指標を取り入れたことで，学習者の理解度を学習者が記述した失敗知識と修正時間の両方から分析することが可能となった．失敗知識の内容だけで理解度を測るのではなく，より実践に近いデータとして理解度を測ることができる．GeneRef 起動有無のスレッショルドを導入することにより，うっかりミスや理解が十分なエラーで発生させた時に内省をさせないよう工夫できた．

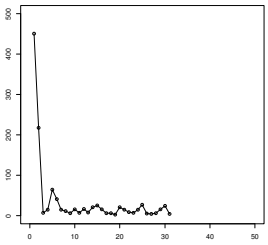
8. おわりに

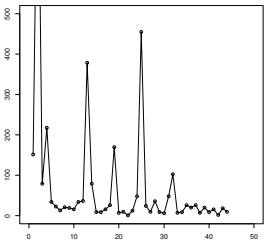
本稿では，修正時間に着目したコンパイルエラーの分析指標を提案し，それを内省学習に適用することで内省学習効果を分析した．その結果，以下の 3 点のことが分かった．1) 失敗知識の内容から理解度の向上が窺える時は，その次の修正時から修正時間が短くなる傾向を持つことが多い．2) 非内省学習者と比較して内省学習者の修正時間は Type1 になりやすい．3) 非内省学習者と比較して内省学習者は Type1 の収束する速度が早い．これらの成果が全て GeneRef の効果であると言い切ることはできないが，修正時間を分析とアプリケーションソフトウェアに導入することによって工夫が可能となった．

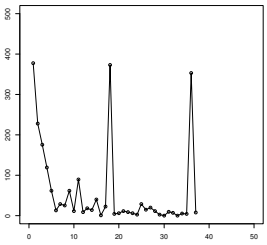
参考文献

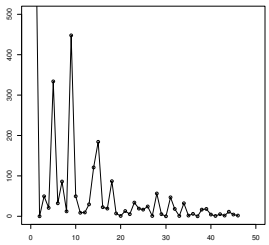
[1] 知見邦彦，櫨山淳雄，宮寺庸造: 失敗知識を利用したプログラミング学習環境の構築，電子情報通信学会論文誌，D-I，J88 DI(1):pp. 66-75(2005).
[2] Matthew C. Jadud: Methods and Tools for Exploring Novice Compilation Behaviour , ICER'06，pp. 73-84(2006).
[3] Guillaume Marceau，Kathi Fisler，Shriram Krishnamurithi: Measuring the Effectiveness of Error Messages Designed for Novice Programmers , ACM Technical Symposium on Computer Science Education，pp. 499-504(2011).
[4] James Jackson, Michael Cobb, Curtis Carver: Identifying Top Java Errors for Novice Programmers , Frontiers in Education 2005, pp. 19-22 (2005).
[5] 梶浦文夫，木村宏: C プログラミング実習における文法エラーの分析，情報処理学会第 47 回全国大会平成 5 年後期(1),pp. 27-28(1993).
[6] 森本康彦，飯島正也，植野真臣，横山節雄，宮寺庸造: プログラミング演習支援のためのコンパイルエラー分析，日本行動計量学会大会発表論文抄録集，Vol.33，pp. 267-268(2005).
[7] Maria Hristova, Ananya Misra, Megan Rutter, Rebecca Mercuri: Identifying and Correctiong Java Programming Errors for Introductory Computer Science Students , SIGCSE '03 Proceedings of the 34th SIGCSE technical symposium on Computer science education, pp. 19-23(2003)
[8] 榊原康友，松澤芳昭，酒井三四郎： プログラミング初学者におけるコンパイルエラー修正時間とその増減速度の分析，情報処理学会シンポジウム論文集，Vol. 4,pp. 121-128(2012).

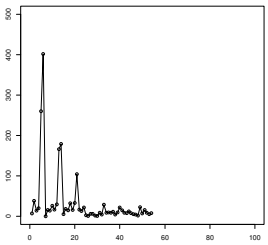

学習者 001(G,Type1)

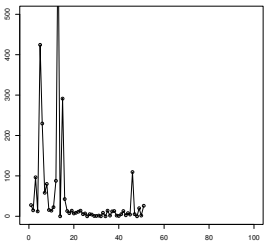

学習者 035(G,Type1)

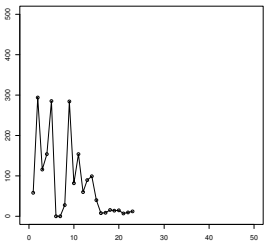

学習者 051(G,Type1)

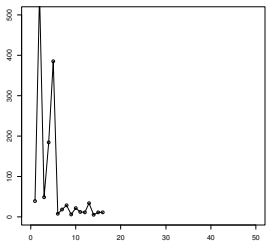

学習者 073(G,Type1)

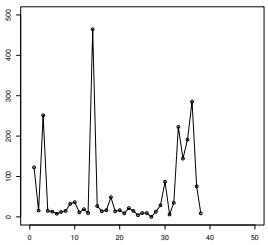

学習者 093(G,Type1)

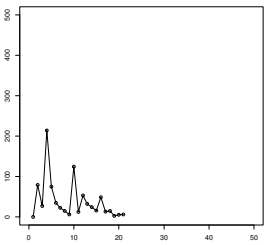

学習者 097(G,Type1)

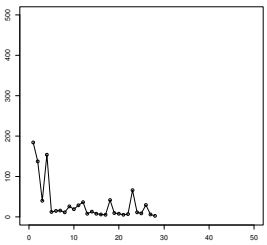

学習者 105(G,Type1)

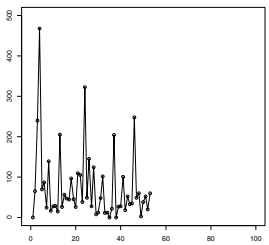

学習者 501(NG,Type1)

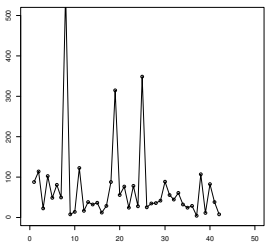

学習者 002(NG,Type1)

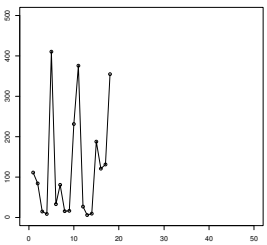

学習者 038(NG,Type1)

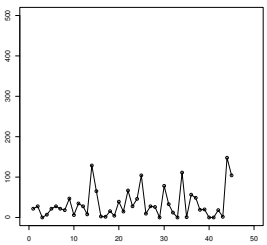

学習者 043(NG,Type1)

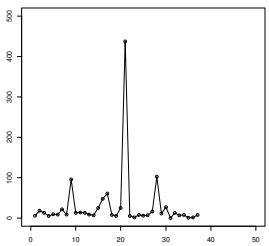

学習者 050(NG,Type1)

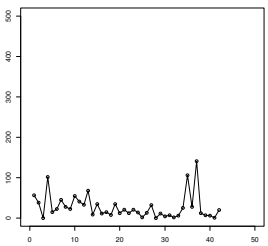

学習者 095(G,Type2)

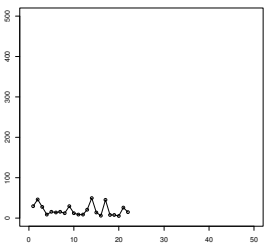

学習者 009(NG,Type2)


学習者 042(NG,Type2)


学習者 071(NG,Type2)


学習者 021(G,Type3)


学習者 055(G,Type3)


学習者 074(NG,Type3)

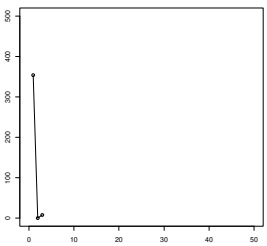
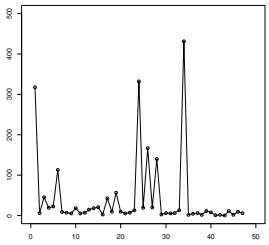
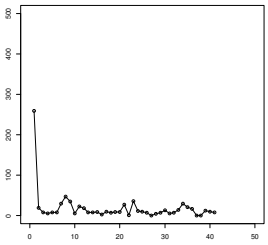
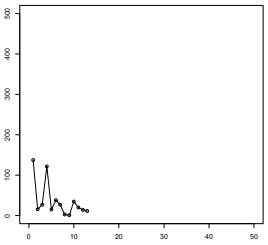
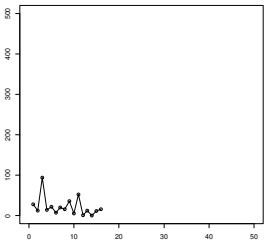

学習者 047(NG,Type4)

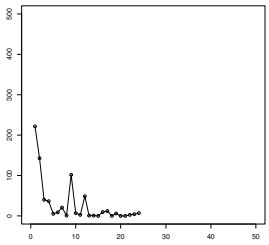
図 8 エラー「構文解析中にファイルの終わりに移りました」における修正時間変化

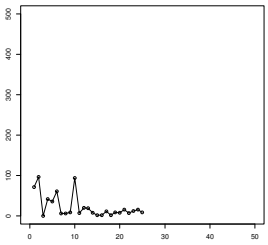

学習者 001(G,Type1)

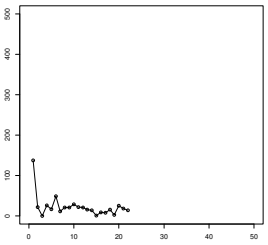

学習者 033(G,Type1)

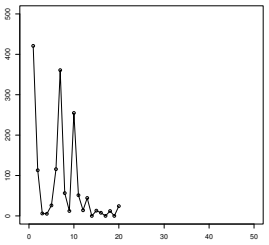

学習者 053(G,Type1)

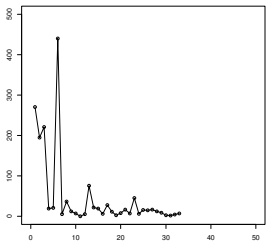

学習者 067(G,Type1)

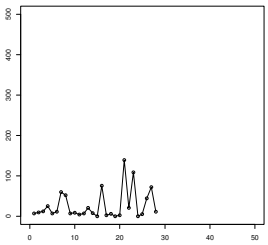

学習者 105(G,Type1)

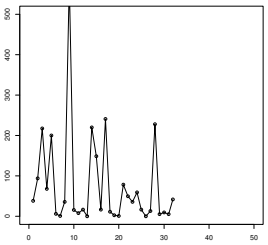

学習者 013(NG,Type1)

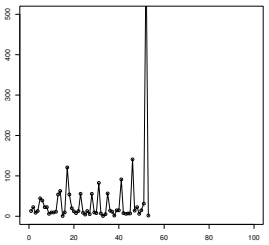

学習者 027(NG,Type1)

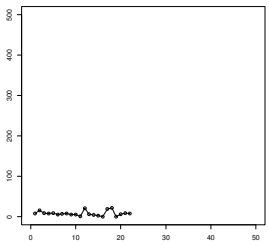

学習者 036(NG,Type1)

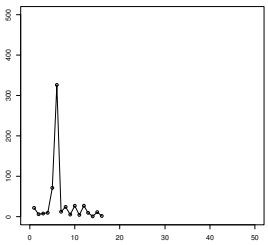

学習者 048(NG,Type1)

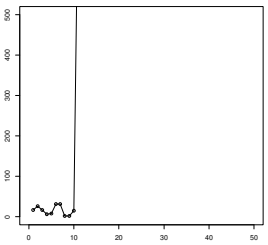

学習者 049(G,Type2)

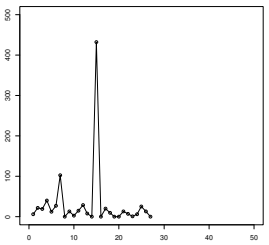

学習者 089(NG,Type2)

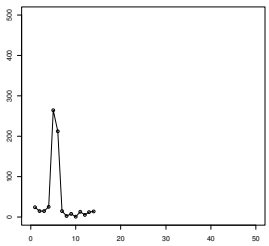

学習者 081(NG,Type2)

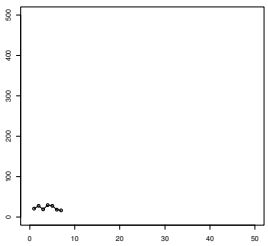

学習者 021(G,Type3)

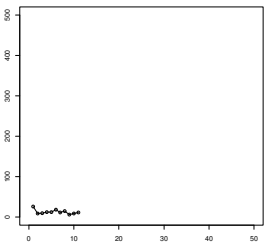

学習者 031(G,Type3)


学習者 039(G,Type3)


学習者 091(G,Type3)


学習者 005(NG,Type3)


学習者 007(NG,Type3)


学習者 094(NG,Type3)

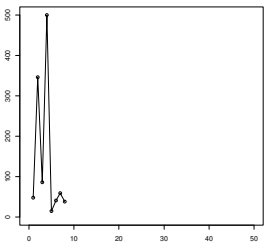

学習者 010(NG,Type4)

図 9 エラー「class,interface または enum がありません」における修正時間変化