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

榊原 康友†, 松澤 芳昭‡, 酒井 三四郎‡

概要

本研究では，プログラミング初学者の学習記録を用い，修正時間とその変化に注目してコンパイルエラーを分析する方法を考案し，実際の記録データに適用しその傾向を明らかにすることを試みた．学習記録から自動で修正時間を算出するために，複合エラーに対応する修正時間の定義を行った．修正時間の変化を捉えるために，時系列修正時間の回帰直線の傾きを算出し，修正時間の増減速度を示す Correction Time Slope（以下，CTS）の定義を行った．Java を用いたプログラミング入門教育の受講生約 100 名が記録した約 82,000 個のコンパイルエラーについて，エラーの種類毎に集計し提案手法を適用した．その結果，以下の知見が得られた．1) 修正時間はべき乗分布になる．2) エラーの種類毎のエラー総数と修正時間の平均には相関がない．3) 構文解析エラーと比較して意味解析エラーの方が修正時間が長い．4) エラーの種類毎の CTS 値について，修正の難易度との相関が示唆される．

An Analysis of the Correction Time of Compile Errors and Its Decreasing Speed for Novice Programmers

Kosuke Sakakibara†, Yoshiaki Matsuzawa‡, Sanshiro Sakai‡

Abstract

In this paper, we proposed a methodology to reveal a trend of a correction time of compile errors and its changing speed for novice programmers by analyzing recorded logs in an introductory programming course. This paper presents two concepts. One is redefinesd the "Correction Time" which supports the automated calculation of the correction time even if the error is one of the combined errors. Another is CTS (Correction Time Slope) which indicates the increasing or decreasing speed of the correction time that is represented by the slope of the regression line calculated by the series of temporal correction time datum. We applied the proposed analysis method for 100 learners' logs recorded in an introductory programming course, the total of 82,000 errors divided into groups by compile error kind. As a result, we have found that 1) distribution of the correction time follows power law, 2) There is no correlation between the number of errors and its correction time for each compile error kind, 3) The correction time for semantic errors are longer than syntax errors, and 4) CTS for each compile error kind indicates the difficulty of the correction.

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．

しかし，教授者がどのエラーに対して学習者に指導するべきかの有用な指標はない．コンパイルエラーの修正難易度を示す指標があれば，指導が必要なエラーと不要なエラーを判断することができる．

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

† 静岡大学大学院情報学研究科
Graduate Schools of Informatics , Shizuoka University
‡ 静岡大学情報学部
Faculty of Informatics , Shizuoka University

ラーだと仮定できる．

学習者のコンパイルエラーの修正に対する理解進行を示す指標はない．コンパイルエラーの修正方法を学習させることにおいて，教授者が学習者の理解進行を把握して指導する必要がある．時間経過による修正時間の変化を指標とすることで，学習者の理解進行を把握できると考えられる．

本研究では，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修正難易度指標の提案と評価を行い，更に，プログラミング初学者から得られたデータに指標を適用した分析を行った．本論文では，コンパイルエラー修正時間の定義，コンパイルエラー修正時間の増減速度の指標の提案と評価，分析結果について述べる．

2 先行研究

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 [1]．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 [2]．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 [3]．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 [4]．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 [5]．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．

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 [6]．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評価しているが，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．

3 提案手法

3.1 目的

本研究ではコンパイルエラー修正時間に着目したコンパイルエラー修正難易度を示す指標の提案と評価を行う．更に，プログラミング初学者のデータに指標を適用し，その傾向を明らかにすることを目的とする．

3.2 修正時間の算出ルール

エラー数と修正時間の基礎データは，プログラミング学習者の学習記録を用い，その学習記録から自動で修正時間を算出するために，複合エラーに対応する修正時間の定義を行った．修正時間のルールは以下の 3 つのルールにまとめられる．図 1 に修正時間の算出モデルを示す．

3.2.1 ルール 1：基本ルール

図 1 の基本ルールの修正時間算出モデルは，セグメントとコンパイルポイントの 2 つの概念で構成される．セグメントはコンパイルの時間間隔を表わしたものであり，モデルでは線分で示されている．コ

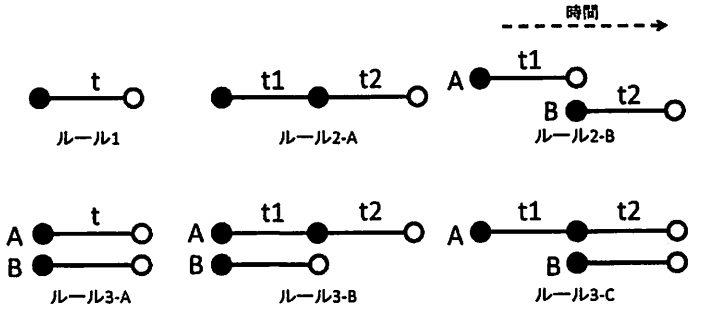

図 1: 修正時間算出モデル

ンパイルポイントは学習者がコンパイルした時点を表わすものであり，エラーが発生している状態かエラーが修正された状態のどちらかの状態を持つ．モデルではセグメントの両端に位置する丸形で示されており，エラーが発生している状態を黒塗り丸形，エラーが修正された状態を白塗り丸形で示している．

ルール 1 は 1 個のみ発生しているエラーを 1 度のコンパイルで修正した場合のルールである．基本ルールの修正時間 T はセグメントを 1 個だけ持つため $T = t$ で表わされる．

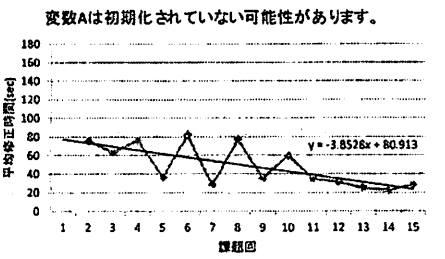

図 2: CTS の算出例

3.2.2 ルール 2 : エラー継続ルール

ルール 2 はコンパイルエラーが 1 度のコンパイルで修正されず，エラーが継続して表れる場合のルールである．図 1 のルール 2 のモデルでは新たにヒストリの概念が追加されている．ヒストリは 1 個のエラーの発生から解消までを表わす概念であり，コンパイルポイントとセグメントを複数持つ．

ルール 2-A は同じエラーメッセージが継続して表れているルールである．このルールではヒストリが持つ全てのセグメントを足し合わせたものを修正時間とする．そこで修正時間 T は $T = t_1 + t_2 + \cdots + t_n$ となる．図 1 の例では修正時間 T は $T = t_1 + t_2$ である．

ルール 2-B は継続ルールでも特殊であり，修正作業によってエラーは修正されなかったがエラーメッセージが変わる場合のルールである．このルールではメッセージの変わったエラーをそれぞれ別のヒストリとして考える．図 1 の例ではそれぞれのエラーの修正時間 T_A，T_B は $T_A = t_1$ ，$T_B = t_2$ である．

3.2.3 ルール 3 : エラー複合ルール

ルール 3 はコンパイルエラーが複合して発生する場合のルールである．

ルール 3-A はエラーが複合する場合の基本ルールである．複数のエラー修正が同じ時間間隔で行われたと考え，セグメントをエラー数で割ったものをそれぞれのエラーの修正時間とする．そのため，ルール 3-A の修正時間は $T_A = T_B = \cdots = T_n = t/n$ で表わされる．図 1 の例では修正時間はそれぞれ $T_A = T_B = t/2$ となる．

ルール 3-B はエラー修正作業によって部分的にエラーが修正された場合のルールである．部分的に修正されたエラーがあった場合，その時間はそのエラーについてのみ修正時間を割り当てたと考える．実際にルール 3-B が適用される場面を 30 抽出した結果，27 の場面で想定通りの修正を行っており，この算出方法は適切だと考えられる．図 1 の例では修正時間はそれぞれ $T_A = t_2$，$T_B = t_1$ で表わされる．

ルール 3-C はエラー修正作業によって新たにエラーが発生した場合のルールである．このルールでは新たなエラーが発生する前の時間は，それまで発生していたエラーの修正時間に割り当てる．図 1 の例では修正時間はそれぞれ $T_A = t_1 + t_2/2$，$T_B = t_2/2$ で表わされる．

3.3 CTS（修正時間増減速度）

学習者のコンパイルエラー修正時間の変化を捉えるために，修正時間の増減速度を示す新しい指標 Correction Time Slope（以下，CTS）を考え，その値のことを CTS 値と名付けた．CTS は各課題回におけるコンパイルエラーの平均修正時間から求めた線形近似式とし，その傾きを CTS 値と定義した．

表 1: 各課題における学習目標

課題	学習目標
1	タートルのメソッドを使う
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	キャスト
5	オブジェクト
6	画像表示，キーボード操作
7	なし（自由課題）
8	ビジュアルプログラミング言語
9	メソッド作成，引数
10	戻り値
11	再帰関数
12	リスト，配列
13	ソート
14	ファイル読み込み，Java 標準ライブラリ
15	なし（自由課題）

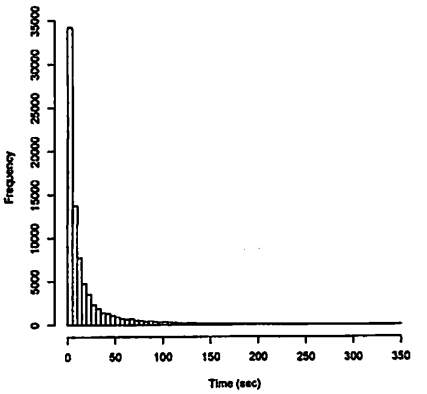

図 3: 修正時間の分布

CTS の算出例を図 2 に示す．グラフの縦軸にエラーの平均修正時間，横軸に課題回をとる．この時，平均修正時間は秒で表わした値，課題回の値は 1 から始めて回を増やす毎に値を 1 ずつ増加させた値とする．図 2 の例では課題回が 15 回あるため 1 から 15 となっている．なお，図 2 の例の課題 1 のようにエラーが発生しておらず，平均修正時間が求まらない回においては空データのままにして CTS を算出する．このデータから線形近似式を求めて得られた傾き-3.85 が図 2 の CTS 値となる．

CTS 値の正負は学習者のエラー修正時間の増減傾向を示す．負の CTS 値はカリキュラムが進むにつれてエラー修正時間が短くなることを示す．学習が起これば言語仕様を理解していくはずであるため，多くのエラーについて CTS 値は負の値となると考えられる．正の CTS 値はカリキュラムが進むにつれて学習者のエラー修正時間が長くなることを示す．学習が起こっていると仮定すれば学習者が理解するよりもカリキュラムの内容が難しくなっていると考えられる．

3.4 分析対象データ

3.4.1 対象授業

エディタの操作履歴データ取得の対象は文科系の大学 1 年生約 100 名である．これらのデータは学生が半期間の Java プログラミング講義 15 回分で課された課題を解く過程で得られたものである．学生は Oracle 社が提供する Java Development Kit に含まれる Javac コンパイラを利用している．各課題では学習目標が設定されており，課題内容がコンパイルエラー数，平均修正時間，CTS 値に影響すると考えられる．そこで表 1 に各課題における学習目標を示す．

3.4.2 データの前処理

取得されたコンパイルエラー総数は 86,452 個，コンパイルエラーの種類（本論文におけるコンパイルエラーの種類とは，シンボル等の変化がある部分を除いたエラーメッセージ 1 個を 1 種類として分類したものを指す）は 75 種類であった．

これらのデータから「学習者がエディタを 5 分以上操作しない時間を含むエラーを除外する」前処理を加える．学習者が数分間，エディタ操作をせずに Web でエラーの修正方法を検索するなどの状況は考えられるが，5 分以上エディタを操作せずにエラーの修正に取り組んでいる状況は想像しにくい．5 分以上操作がない状況とは，学習者が席を外すなどの理由が考えられ，学習者がそのエラーの修正に取り組んだ時間とは異なると考えられるためデータから除外した．

更に，CTS を算出する場合，「各種のエラーにおいてエラー数が 3 個以下の課題回があった場合，その課題回のデータ」を除外する．あるエラーの種類においてエラー数が極端に少ない課題回があった場合，平均修正時間が少ないサンプルによって大きく影響を受けてしまう．その影響により CTS を求める際に絶対値が極端に大きな CTS 値が算出されてしまうことで分析できなくなることを避けるためである．

4 結果

4.1 修正時間の全体的傾向

データ前処理後のコンパイルエラー総数は 81,800 個，エラーの種類は 75 種類で，そのうち 41 種類が意味解析エラー，構文解析エラーが 34 種類であった．エラー 1 個の平均修正時間は 27.5 秒であった．算出されたエラー毎の指標一覧を付録の表 2 に付す．

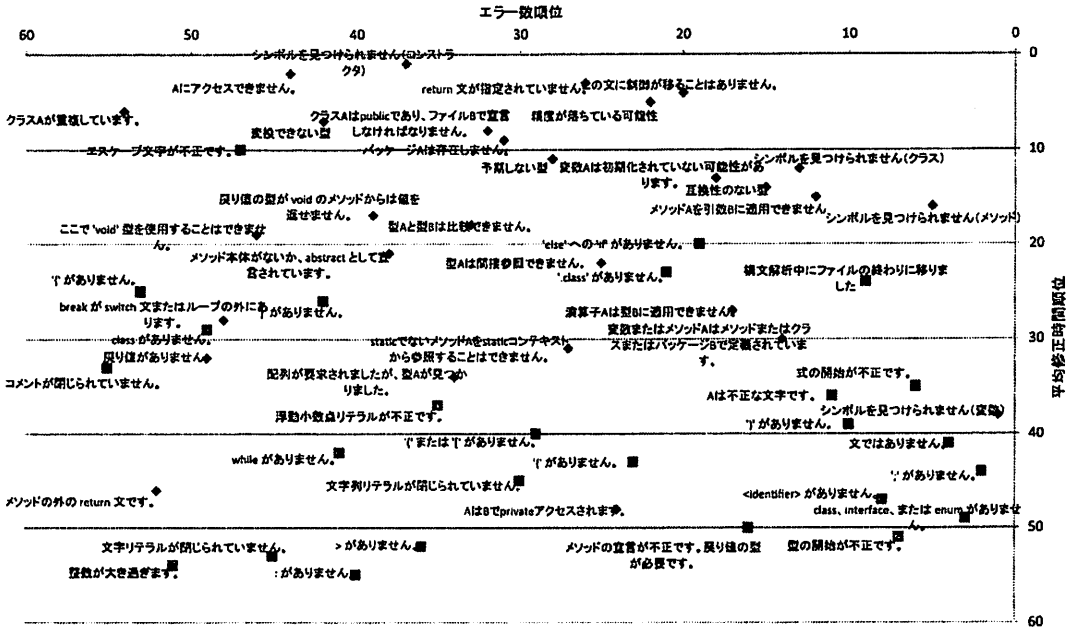

図 4: エラー数順位と平均修正時間順位の相関

エラー修正時間のヒストグラムを図 3 に示す．図 3 よりエラー修正時間はべき分布になることが分かった．

べき分布の平均値は正規分布の平均値とは意味が異なる．本論文ではエラー修正時間の平均値は主にそのエラーにおける修正が困難であった場面の修正時間であると解釈した．これはべき分布の平均値がサンプルの選び方によって大きく変動するためで，修正時間の平均の場合に大きく影響されるのは修正時間の長いサンプルだからである．

4.2 エラー数と平均修正時間の相関

エラー数 10 個以上のコンパイルエラー 55 種類*に対しエラー数と平均修正時間の順位を算出し，エラー数の順位を横軸，平均修正時間の順位を縦軸にプロットしたものが図 4 である．このプロットでは順位が高いほどエラー数が多く，修正時間が長いことを示している．また，プロットの赤色の点が意味解析エラー，青色の点が構文解析エラーを示している．

図 4 よりエラー数と平均修正時間の相関がないことが示された．学習者にとって修正が難しいエラーは図 4 中の右上に分布するエラーである．学習者が目にする機会が多いにもかかわらず修正時間が長いエラーであるため，学習者にとってこれらのエラーは他のエラーよりも修正が難しいエラーと考えられる．

図 4 より意味解析エラーが平均修正時間の上位，構文解析エラーが下位に分布していることが分かる．このことから，構文解析エラーと比較して意味解析エラーの方が修正時間が長いことを客観的に示すことができた．実際の修正時間では意味解析エラーの平均修正時間は 36.7 秒，構文解析エラーの平均修正時間は 21.3 秒であった．

4.3 修正時間の増減速度

各種のエラーに対して CTS の分析を行った．前処理後に，課題回 1 回分でしか発生していないエラーが 27 種類あったため，残りの 48 種類のエラーに対して CTS の分析を行った．各種のエラーの CTS 値は正の値のエラーが 17 種類，負の値のエラーが 31 種類であった．

4.3.1 CTS 値が負の値をとるコンパイルエラー

CTS 値が負の値のエラー例を図 5 に示す．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」は図 5 左部から課題回を経ることで平均修正時間が短くなっている様子が分かる．このような特徴を持つエラーは学習者自身で修正時間を短くす

*本稿ではエラーメッセージに A,B の記号が含まれるものがある．これらのエラーメッセージはシンボルなどの変化する部分を含むものであり，それらを同一のエラーメッセージとして表記するために記号に置き換えている．

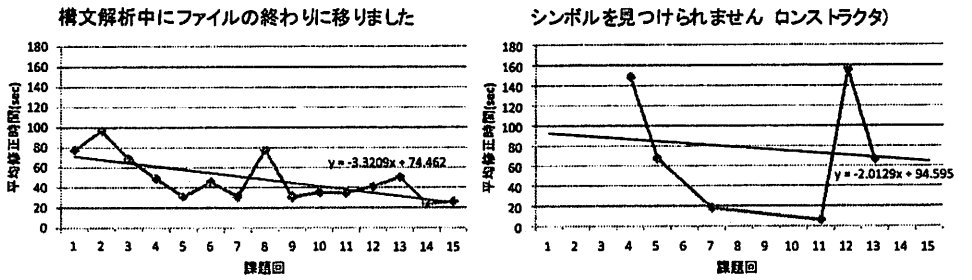

図 5: 負の CTS 値のコンパイルエラー例

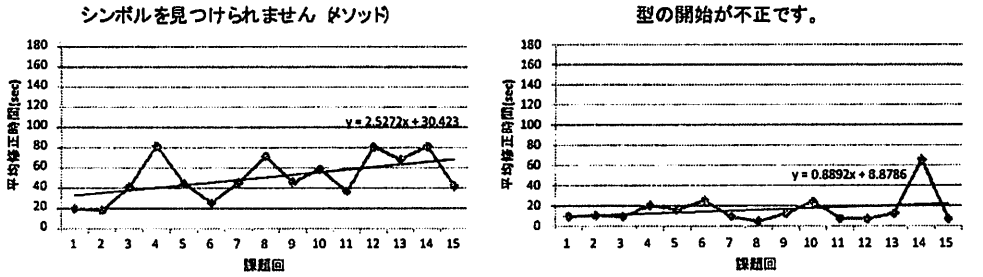

図 6: 正の CTS 値のコンパイルエラー例

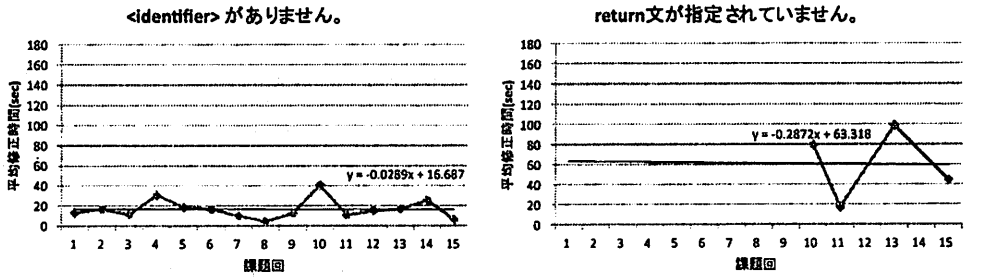

図 7: CTS 値が 0 に近いコンパイルエラー例

ることができるため，学習者に対する特別な指導は必要ないと考えられる．

エラーメッセージ「シンボルを見つけられません (コンストラクタ)」は CTS 値が-2.0 であるが，図 5 右部を見ると時間が非常に長い課題回や非常に短い課題回がありバラつきが大きく，この変化からは学習者の修正時間が改善されているとは言えない．

CTS 値が最小値のエラーはエラーメッセージ「エスケープ文字が不正です」であり，CTS 値が-73.2 と絶対値が非常に大きな数値を算出している．その原因としてはこのエラーが全 15 回の課題の中で課題回 2 回分しか発生してないことが挙げられる．その他のエラーで CTS 値の絶対値が 10 に近いものを見ると 2 回や 4 回と発生頻度が少ないエラーであった．

4.3.2 CTS 値が正の値をとるコンパイルエラー

CTS 値が正の値のエラー例を図 6 に示す．エラーメッセージ「シンボルを見つけられません (メソッド)」は図 6 左部から課題回を経るうちに平均修正時間が長くなっている様子が分かる．原因はカリキュラムが進むにつれて新しいメソッドが増加していき，学習者がそれらのメソッドを使いこなすのに時間が

かかったためだと考えられる．

図6右部のエラーメッセージ「型の開始が不正です。」のようにほとんどの課題回は修正時間が短いが，課題回終盤で数回だけ修正時間が長くなっているエラーがある．このようなエラーは平均修正時間が短い構文解析エラーによく見られた．

4.3.3 CTS値が0に近い値をとるコンパイルエラー

CTS値が0に近い値のエラー例を図7に示す．エラーメッセージ「<identifier>がありません。」は図7左部から課題の序盤から終盤まで修正時間が短いことが分かる．修正時間が更に短くなる傾向は見られないが，このようなエラーは学習者にとって修正が容易であると考えられる．

エラーメッセージ「return文が指定されていません。」はCTS値が0に近い値だが図7右部から平均修正時間の変化にばらつきが生じていることが分かる．このようなエラーはCTS値が0に近い値のエラーの中でも平均修正時間が長い意味解析エラーによく見られる．この傾向は課題内容に影響されるために生じるか，もしくは言語仕様を理解していない学生がいるために生じると考えられる．

4.3.4 エラー全体のCTS分析

エラー全体についてのCTS値を求めると0.02であり0に近い結果であった．これはエラー全体の修正時間の変化がほとんどないことを示すが，各種エラーのCTS値を見ると負の値のエラーが多い．実際に初回の課題で発生された17種類のエラーのうち10種類のエラーのCTS値が負の値であった．

これにより，エラーの修正時間が短くなっていないのではなく，カリキュラムを進めることによって新たに発生したコンパイルエラーがエラー全体のCTS値に影響していると考えられる．エラー全体のCTS値は新しいコンパイルエラーの頻度や難易度が関係するため，カリキュラムの進行具合と学生の言語仕様の理解の速度がエラー全体のCTS値となると推測される．

5 考察

5.1 エラー数と平均修正時間の相関の考察

図4上部，特に右上に分布するエラーは指導する必要があるエラーだと考えられる．キャスト関連のエラーが上部に位置する典型的なエラーである．

しかしながら，図4上部に分布するエラーであっても必ずしも指導が必要なエラーとは言えない．この図4では相対的な評価をしているため，元々修正に時間がかかるエラー，時間がかからないエラーを考慮していない．そこで，CTSなどの他の指標と組み合わせることによって，指導の必要なエラーがより明確に判明すると考えられる．

5.2 CTSの考察

一部の各種コンパイルエラーについてCTS値と修正難易度との相関を示唆する結果が得られた．エラー数の多いコンパイルエラーはCTSと平均修正時間の変化が類似しているものが多い．学習者が目にする機会が多く，極端に修正時間が長いサンプルが少ないからだと考えられる．

しかし，CTS値だけでエラー修正難易度を判定することはできない．その理由は図5のようにCTSとプロットが類似しないエラーがあるからである．類似しない理由としては以下の3点が考えられる．1) 修正時間が課題内容に影響される．2) エラー数が少ない回がある場合，サンプルに影響される．3) 修正時間が極端に長いサンプルの影響．

修正時間が課題内容に影響される場合，その課題回で平均修正時間が長くなる．これはCTSの算出はエラーの種類毎で行っており，原因まで把握することができないのが理由である．

極端なサンプルの影響を抑える必要がある．べき分布での平均修正時間はサンプルに影響されやすいため，影響の少ない中央値でCTSを算出し，妥当性の比較をするのが今後の課題である．

参考文献

[1] Matthew C. Jadud: Methods and Tools for Exploring Novice Compilation Behaviour, ICER'06, pp. 73-84(2006).

[2] 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).

[3] James Jackson, Michael Cobb, Curtis Carver: Identifying Top Java Errors for Novice Programmers, Frontiers in Education 2005, pp. 19-22 (2005).

[4] 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).

[5] 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).

[6] 知見邦彦，樋山淳雄，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).

付録

A　エラーの種類毎の指標一覧

エラーの種類毎の指標一覧を表 2 に付す．表 2 中，CT とは，エラー修正時間 (Correction Time) の平均値を示す．

表 2: エラーの種類毎の指標一覧 (CTS 順)

ID	メッセージ	数	CT	CTS
1	エスケープ文字が不正です。	27	55.8	-73.2
101	A にアクセスできません。	34	77.1	-14.3
102	変換できない型	36	62.5	-9.2
2	文字リテラルが閉じられていません。	33	10.4	-7.8
103	この文に制御が移ることはありません。	578	71.5	-5.0
104	変数 A は初期化されていない可能性があります。	887	53.9	-3.9
3	構文解析中にファイルの終わりに移りました	2959	42.8	-3.3
4	予期しない型	158	54.4	-3.2
105	変数またはメソッド A はメソッドまたはクラスまたはパッケージ B で定義されています。	1788	33.9	-3.1
5	'.class' がありません。	533	42.9	-2.9
6	整数が大き過ぎます。	18	8.2	-2.9
106	戻り値がありません。	19	29.9	-2.6
107	シンボルを見つけられません（クラス）	1863	53.9	-2.2
108	シンボルを見つけられません（コンストラクタ）	93	92.8	-2.0
7	'else' への 'if' がありません。	653	46.1	-1.8
8	浮動小数点リテラルが不正です。	97	25.5	-1.5
9	文字列リテラルが閉じられていません。	135	19.8	-1.4
109	精度が落ちている可能性	508	64.1	-1.1
110	互換性のない型	1308	51.7	-1.1
111	戻り値の型が void のメソッドからは値を返せません。	54	49.0	-1.0
112	static でないメソッド A を static コンテキストから参照することはできません。	176	30.6	-1.0
10	式の開始が不正です。	4387	26.1	-0.9
11	'}' がありません。	36	39.2	-0.9
12	class、interface、または enum がありません。	8072	13.4	-0.8
13	A は不正な文字です。	2202	25.7	-0.8
14	文ではありません。	4596	21.7	-0.6
15	')' がありません。	2252	23.2	-0.5
16	: がありません。	53	4.0	-0.4
113	型 A と型 B は比較できません。	105	46.6	-0.4
114	return 文が指定されていません。	207	72.7	-0.3
115	クラス A は public であり、ファイル B で宣言しなければなりません。	126	57.5	-0.2
17	<identifier> がありません。	3084	17.5	0.0
116	型 A は間接参照できません。	244	44.0	0.1
18	'(' がありません。	414	21.1	0.1
19	';' がありません。	13791	20.5	0.5
117	シンボルを見つけられません（変数）	16397	24.0	0.6
20	> がありません。	95	11.7	0.7
118	演算子 A は型 B に適用できません。	1143	38.3	0.7
119	メソッド A を引数 B に適用できません	2192	50.4	0.8
21	型の開始が不正です。	3676	12.8	0.9
120	A は B で private アクセスされます。	385	17.4	1.1
22	メソッドの宣言が不正です。戻り値の型が必要です。	1285	13.0	1.2
121	パッケージ A は存在しません。	134	56.4	1.7
23	'(' または '[' がありません。	152	22.0	1.8
122	メソッド本体がないか、abstract として宣言されています。	83	46.1	2.1
123	シンボルを見つけられません（メソッド）	4409	50.4	2.5
124	ここで 'void' 型を使用することはできません。	30	46.1	3.9
125	break が switch 文またはループの外にあります。	26	38.1	9.3