Compile Error Collection Viewer: 修正履歴分析によるコンパイルエラー学習支援システム

平尾 元紀[1,a)]　松澤 芳昭[2,b)]　酒井 三四郎[2,c)]

概要：本研究では，学習者が個人のコンパイルエラー修正履歴を観察，分析できるシステム CocoViewer(Compile error Collection Viewer) を開発した．本システムは，プログラミングの授業でコンピュータに蓄積された開発環境の操作履歴を利用してコンパイルエラーの修正時間を計算し，その推移グラフを自動生成し提示する．学習者はエラーの種類毎に集計された推移グラフの一覧や，各個のエラーが生じた状況の詳細な分析が可能である．これまで学習者各々が漠然と抱いていたコンパイルエラー修正の学習状況が定量的に把握されることで，学習の動機付け，プロセスの改善促進，およびエラーに対する恐怖感の軽減をすることが本研究の目的である．文科系の大学生を対象としたプログラミング入門教育の受講者約 100 名に試用実験を行った．その結果，被験者は興味を持ってシステムを利用し，エラーに対する対話が促進され，コンパイルエラーについての誤った認識が改善し，コンパイルエラーへの恐怖感が軽減されるという結果が得られた．

Compile Error Collection Viewer: Learning Support System for Compilation Error by Analyzing Error Fixing Records

Hirao Motoki[1,a)]　Matsuzawa Yoshiaki[2,b)]　Sakai Sanshiro[2,c)]

Abstract: We have developed CocoViewer (Compile error Collection Viewer) which was designed for learners in programming to enable them to conduct an analysis for their compiling error records. CocoViewer generates charts that show a trajectory of reducing the fixing time of the compilation error that is calculated by all logs recorded in students' computer during a programming course. Students can see lists of charts for all kinds of compilation error, as well as can see a particular detailed circumstance of error that is selected by a student. We hypothesized that the system promotes clear understandings regarding their compilation error learning, and following three effects were expected: (1) to encourage more experiences of compilation error fixing, (2) improvement of the procedure of fixing error, and (3) reducing unarticulated anxiety for the compilation error. The system was tried in an undergraduate introductory programming course for approximately 100 non-CS, liberal art students. The both qualitative research how students use the system and quantitative data in questionnaire showed that the students appreciated the system and enjoy to use, the system promoted dialogues among students regarding compilation errors, and we succeeded to reduce unarticulated anxiety for students.

1. はじめに

Java を利用したプログラミング言語学習では，コンパイルエラーの発生は避けられない．学習者にとってコンパイルエラー修正は難しいものである．学習者がコンパイルエラー修正を難しいと思いこむことは，学習者がコンパイルエラーに対して恐怖感を持つ原因の一つである．コンパイルエラーに対して恐怖感を持っている学習者は，コンパイルエラーを出すことを避けるべきであると誤った認識をしがちである．

しかし，コンパイルエラーは文法上のミスを発見できる

[1] 静岡大学情報学研究科
Graduate Schools of Informatics , Shizuoka University
[2] 静岡大学情報学部
Faculty of Informatics , Shizuoka University
[a)] hirao@sakailab.info
[b)] matsuzawa@inf.shizuoka.ac.jp
[c)] sakai@inf.shizuoka.ac.jp

有用な情報である．コンパイルエラーは発生しても早く正確に修正すればよい．学習者が早く正確にコンパイルエラーを修正するためには，コンパイルエラー修正の学習の促進が必要である．

コンパイルエラー修正の学習を促進するためには，学習者がコンパイルエラーに対して誤った認識を改善し，コンパイルエラーを出して修正することが必要である．コンパイルエラーに対する誤った認識の改善のためには，コンパイルエラーに対する恐怖感を軽減することで実現できると考えられる．

そこで本研究では，学習者がコンパイルエラー修正を早く正確に行うために，以下の三点を目的とする．

- コンパイルエラー修正プロセスの改善
- コンパイルエラー学習の動機付け
- コンパイルエラーに対する恐怖感の軽減

これらの目標を達成するためには，学習者が自らの修正してきたコンパイルエラーを分析し，学習状況を把握することで実現できると考えた．

本研究では，学習者が個人のコンパイルエラー修正履歴を観察・分析できるシステムを開発した．システムの効果を図るため，文科系のプログラミング学習者に試用実験を行った．

2. コンパイルエラーに対する意識調査

2.1 調査方法

2013 年度に静岡大学情報学部情報社会学科 1 年次で開講された Java を用いたプログラミング入門講義の第 9 回時にアンケートを実施した．文科系の大学生 97 名がアンケートに回答した．

2.2 調査結果

全体の作業に対するコンパイルエラー修正時間の割合のイメージについて聞いたところ，平均 28.4%という回答であった．筆者による実測では平均 15%前後であった．学習者のコンパイルエラーに対して抱いているイメージと実際のコンパイルエラー修正の状況に差異があることがわかる．

コンパイルエラーに対し恐怖感を持っているかどうかを聞いたところ，学習者の 62%がコンパイルエラーに対して少なくともやや恐怖感を持っていると答えた．

この結果から，学習者はコンパイルエラー修正に対し漠然としたイメージを持つことしかできず，コンパイルエラーに対し誤った認識を持っていると考えられる．

3. 先行研究

榊原らはコンパイルエラー学習を支援するシステム GeneRef を考案している [1]．GeneRef はコンパイルエラーを修正した際，学習者が発生原因・修正方法を内省を記述することで，学習者のコンパイルエラーに対する理解を促進する．榊原らは真面目に内省した学習者のコンパイルエラー修正時間は短くなる傾向であることを示唆した．

山本らは事例ベースを用いた動的な学習手法を取り入れた C 言語学習支援システムを考案している [2]．山本らのシステムでは，学習者がブラウザ上でコンパイルエラーと論理エラーを解決した際に，エラー要因を学習者が記述することで，学習者がエラーを発見することを助けている．

高橋は三重大学の講義で用いられているシステム PROPEL を拡張し，受講者が発生させた構文エラー，微軽なエラー，動作エラーをブラウザ上でタイムチャートを表示可能な機能を考案している [3]．この機能により，講師はエラーの修正に手間取っている受講者を発見できる．

小島は PROPEL を拡張し，受講者の製作途中のプログラムを解析し，構文に関するエラーについてアドバイスを表示する機能を考案している [4]．この機能により，講師が多数の受講者に対して効率のよいアドバイスが可能である．

Björn らはコンパイルエラー修正を手助けるシステム HelpMeOut を考案している [5]．HelpMeOut は修正したコンパイルエラーのソースコード情報を収集し，利用者がコンパイルエラーに直面した時に修正提案として，過去に同じコンパイルエラーに直面した他人の修正したソースコードを表示し，コンパイルエラー修正を助ける．

関連研究 [1][2] では，学習者が修正回数を経るごとにどの程度コンパイルエラー修正プロセスが改善されたのかわからない問題がある．学習者はコンパイルエラー修正ができていないと思い込んでいる可能性があると考えられる．

関連研究 [3] では，可視化されたエラー情報を閲覧できる者が講師に限られる問題がある．コンパイルエラーに対して誤った認識を持った学習者がコンパイルエラーを避けてプログラミングをした場合，講師はエラー修正に対して適切なアドバイスが出来ないと考えられる．

関連研究 [4][5] では，システムによりコンパイルエラー修正を助けすぎてしまう問題がある．システムに頼りきった学習者はコンパイルエラー修正を身につけられない．

4. 提案システム

4.1 システムの設計

本研究では榊原ら [6] が定義したコンパイルエラー修正時間の推移を折れ線グラフ（修正時間推移グラフ）として自動生成し，学習者が自らのコンパイルエラー修正を分析できるシステムの開発を行う．システムの名称は収集したコンパイルエラー修正情報を提示することから，CocoViewer(Compile error Collection Viewer) と名付けた．

コンパイルエラー修正情報の分析及び収集は榊原ら [6] のコンパイルエラー分析技術により実現している．コンパイルエラー修正情報収集の流れを**図 1** に示す．コンパイルエラー分析技術の概要は，プログラミング開発環境から自

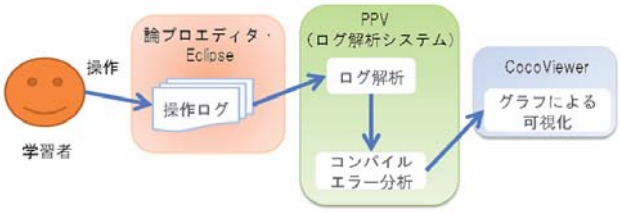

図 1　コンパイルエラー修正情報の収集

動的に得られた操作履歴を利用し，Programming Process Visualizer(PPV)[7] で解析することにより実現する．コンパイルエラー修正時間はこの分析過程において，榊原ら [6] の提案手法に基づき計算されている．

本研究での実装部分は CocoViewer の修正時間推移グラフによる可視化部分である．CocoViewer では分析されたコンパイルエラー情報から自動的に修正時間推移グラフを生成し，学習者に提示する．

4.2　システム動作環境

システムの動作環境は以下のとおりである．

対応言語　Java

コンパイラ　Java Development Kit(JDK) 1.7.0_45

開発環境　論プロエディタ [8]，Eclipse

4.3　修正時間推移グラフ

修正時間推移グラフの例を**図 2** に示す．修正時間推移グラフは学習者個人のコンパイルエラーを対象とする．修正時間推移グラフは X 軸をコンパイルエラー修正回数（単位：回目），Y 軸をコンパイルエラー修正時間（単位：秒）とし，コンパイルエラーの種類 1 個につき 1 つ自動生成する．

図 2 に示す例では，このコンパイルエラーの修正に一回目に 204 秒かかり，二回目に 426 秒かかったことが読み取れる．コンパイルエラーの種類 1 個は，コンパイルエラーメッセージ 1 個と対応している．例えば「';' がありません」がコンパイルエラーの種類の 1 つとして挙げられる．シンボルなど一部分が変化するエラーメッセージの場合は抽象化して同じコンパイルエラーの種類として扱う．

学習者は自らの修正時間推移グラフの概形を分析することで，自らのコンパイルエラー修正の分析が可能である．

4.4　CocoViewer メインウィンドウ

本システムを起動すると，**図 3** に示す CocoViewer メインウィンドウが表示される．CocoViewer メインウィンドウでは修正時間推移グラフ一覧表と，全体のコンパイルエラー修正に関する情報が確認できる．コンパイルエラー修正状況は学習者による個人差があるため，学習者ごとに異なる修正時間推移グラフが表示される．

CocoViewer メインウィンドウの特徴は修正時間一覧グラフを一覧できることである．この特長により，これまで経験したコンパイルエラーの種類がいくつか，どんな修正時間推移グラフの概形が多いのか，ということを学習者がひと目で確認することができる．

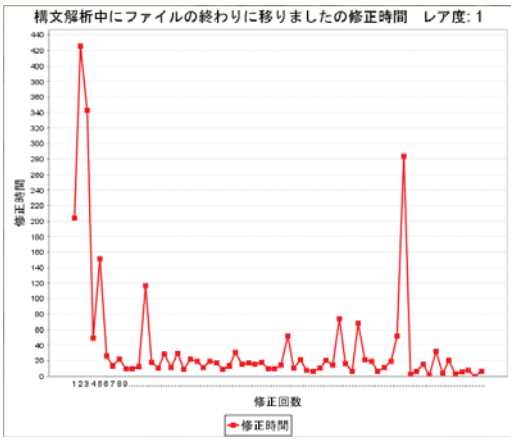

図 2　修正時間推移グラフの例

加えて，学習者が互いに本システムを比較しあうことを想定し，次のデザインを採用している．

- コンパイルエラーの種類にレア度を設定
- レア度に対応した修正時間推移グラフの背景色を設定
- 未発生のコンパイルエラーの種類は，発生状況のヒントとしてコンパイルエラーメッセージを表示

このデザインにより本システムを図鑑として見ることができ，学習者同士が積極的に互いのコンパイルエラー修正について分析することを狙いとしている．

レア度はコンパイルエラーの種類の発生頻度を表している．コンパイルエラーの種類の発生頻度は，榊原ら [6] が集計した 2010 年度に文科系の大学 1 年生向けに開講された Java を用いたプログラミング入門講義の受講者約 100 名のコンパイルエラー発生頻度を用いた．

4.5　修正詳細ウィンドウ

CocoViewer メインウィンドウに表示されているグラフをクリックすると，修正詳細ウィンドウが開く．**図 4** に例を示す．図 4 は「互換性のない型」についての修正詳細ウィンドウである．修正詳細ウィンドウは，拡大された修正時間推移グラフと修正詳細テーブルからなる．収載詳細テーブルでは修正時間・修正日時・コンパイルエラーを修正したプログラム名・修正時間を確認可能である．

4.6　ソースコード比較ウィンドウ

詳細情報テーブルの行をクリックすると，ソースコード比較画面ウィンドウが表示される．**図 5** に例を示す．図 5 は，図 4 の 4 行目をクリックした際のソースコード比較ウィンドウである．ソースコード比較ウィンドウではログ解析システムの情報から，修正情報テーブルで選択した行の時刻時点でのソースコードを再現する．左側がそのコン

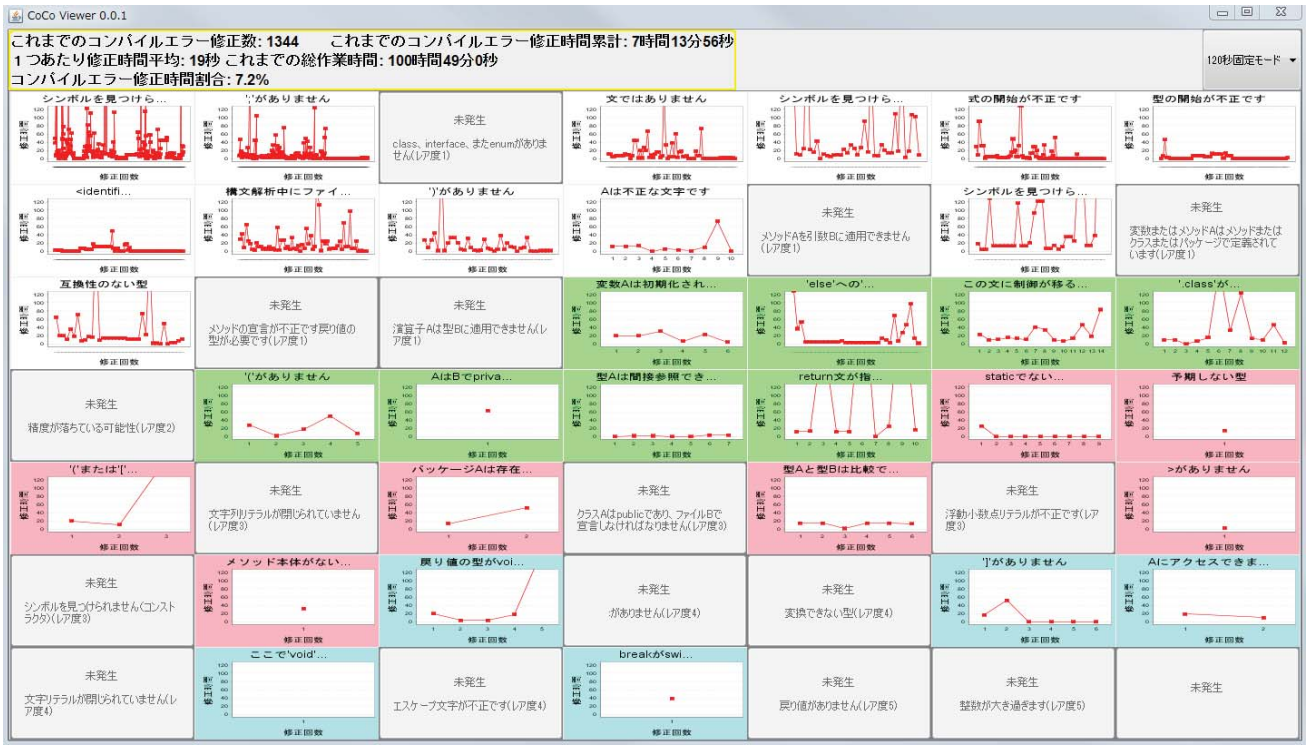

図 3　CocoViewer メインウィンドウ

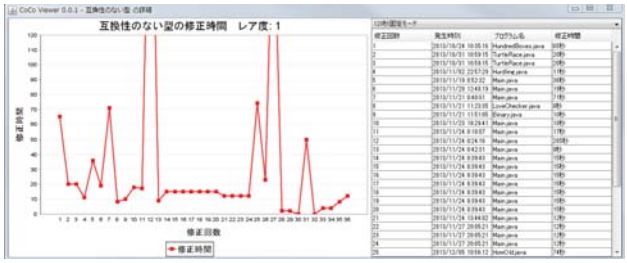

図 4　修正詳細ウィンドウ

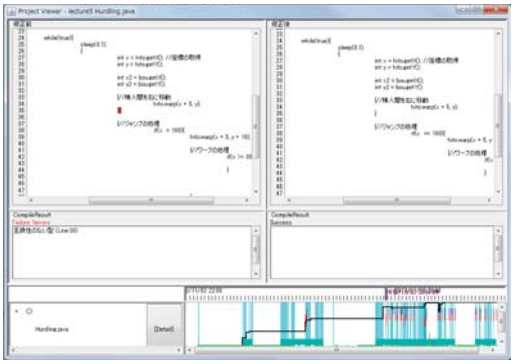

図 5　ソースコード比較ウィンドウ

パイルエラー修正する前のソースコード，右側がそのコンパイルエラー修正した後のソースコードである．図 5 ではコンパイルエラーが発生した 2013/11/02 22:57:29 時点でのソースコードを，コンパイルエラーを修正する前のソースコードとしている．学習者はコンパイルエラー修正前後のソースコードを比較することでコンパイルエラーの発生原因，修正方法を詳細に分析できる．

下部のタイムラインのバーを移動させることにより，コンパイルエラー修正前・コンパイルエラー修正後のソースコードが変化する．この変化により，学習者が行ったエディタ操作を再現できる．学習者は再現されたエディタ操作から，学習者自身がコンパイルエラーを修正する際に行った思考プロセスをなぞり，コンパイルエラーの発生原因，修正方法を考察できる．

4.7　想定される学習者のグラフ読み取り

学習者は修正時間推移グラフの概形を分析することにより，自らのコンパイルエラー修正を分析できる．修正時間グラフの概形は右下がり型，振動型，右上がり型の 3 パターンに分類できる．

4.7.1　右下がり型修正時間推移グラフ

図 6 に右下がり型修正時間推移グラフ（以下，右下がり型）の例を示す．右下がり型では修正回数が増えるに連れ修正時間が短くなっていることから，学習者はこのコンパイルエラーの種類の修正を克服し，理解していると読み取る．順調に学習が進んでいる学習者の場合，修正回数が増えるごとに修正時間が短くなると考えられ，右下がり型が多くなると予想される．学習者にとって右下がり型が理想的な形である．

4.7.2　振動型修正時間推移グラフ

図 7 に振動型修正時間推移グラフ（以下，振動型）の例を示す．振動型では修正回数に関わらず，修正時間が大きく変動することから，学習者はこのコンパイルエラーの種

図 6　右下がり型修正時間推移グラフ

図 7　振動型修正時間推移グラフ

図 8　右上がり型修正時間推移グラフ

類の修正について理解しつつある段階であると読み取る．学習者は振動型のコンパイルエラーについて発生原因と修正方法について学習することで振動型を右下がり型とすることができる．

4.7.3　右上がり型修正時間推移グラフ

図 8 に右上がり型修正時間推移グラフ（以下，右上がり型）の例を示す．右上がり型では修正回数が増えるに連れ，修正時間が増えていくことから，学習者はこのコンパイルエラーの種類の修正について理解できていないと読み取る．右上がり型が現れる要因として，学習者がプログラムの処理を理解するので手一杯であり文法の理解まで出来ないこと，複数のクラスにまたがる修正が必要なコンパイルエラーであることが挙げられる．

4.8　想定されるコンパイルエラー軽減プロセス

以下のプロセスにより，本システムの利用者はコンパイルエラーに対する恐怖感が軽減される．

(1) 修正時間推移グラフを見て，グラフの概形を分析する．
(2) コンパイルエラー修正に対して抱いているイメージと実際の修正状況の差異を埋める．
(3) 自分自身のコンパイルエラー修正の学習状況を認識する．
(4) コンパイルエラーに対する恐怖感が軽減される．

5.　実験計画

5.1　仮説

試用実験の仮説は以下のとおりである．

仮説　CocoViewer を用いると，コンパイルエラー修正に対する誤った認識が改善され，学習者のコンパイルエラーに対する恐怖感が軽減される

5.2　実験環境

2013 年度に静岡大学情報学部情報社会学科 1 年次で開講された Java を用いたプログラミング入門教育の講義（全十五回）を履修している文科系の大学生約 100 名を対象とする．被験者のほとんどが本講義以前にプログラミングの学習経験のない学習者である．

5.3　実験概要

第 12 回講義時に，必須ではない加点課題として本システムの試用実験を行った．実験の手順は以下のようになる．

(1) 被験者にワークシートを配布する．
(2) 講義開始時に，筆者から修正時間推移グラフの読み取り方，実験概要について説明する．(10 分)
(3) 被験者はワークシートにそって，本システムを操作し，ワークシートの質問に記述する (10-15 分)
(4) 講義終了時にワークシートを回収する．

本講義は 8:40 から 11:50 まで開講されている．筆者による説明の後は講義時間内である限り，いつワークシートに取り組んでも良いこととした．

被験者からのワークシート，本システムの操作方法に関する質問は随時受け付けた．筆者は実験中，被験者のシステム使用状況を観察・記録した．

ワークシートの概要は以下のとおりである．

(1) システム利用前アンケート (4 項目)
(2) システムの起動方法，修正時間推移グラフの読み取り方，レア度についての説明
(3) CocoViewer メイン画面を見ながら回答するアンケート (11 項目)
(4) 近くにいる人同士で CocoViewer メイン画面を見せ合いながら回答するアンケート (2 項目)
(5) この実験を通しての振り返りアンケート (4 項目)

5.4　分析方法

自由記述欄を除き，ワークシートのアンケートに回答漏れがある場合，正しく実験が行えていないと判断し，被験者から除外した．

筆者がワークシートを分析したところ，有効と判断した

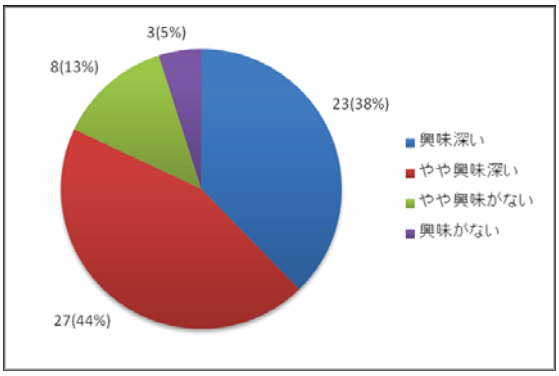

図 9　「修正時間推移グラフ一覧表を見て，興味深いと思いましたか？」の回答結果

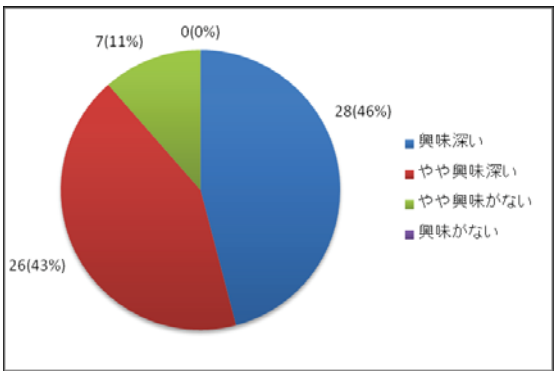

図 10　「友達の修正時間推移グラフ一覧表と自分のものを比較して，興味深いと思いましたか？」の回答結果

被験者は 61 名であった．ワークシートの提出者は 71 名であった．

6.　実験結果

図 9 は自分のコンパイルエラー修正履歴が興味深いかどうかを聞いた質問の回答である．被験者の 82%が少なくともやや興味を持っていると答えた．被験者にとって自らのコンパイルエラー修正履歴を定量的・定性的に振り返る機会がなかったため，被験者が興味を示したと考えられる．

図 10 は他の人のコンパイルエラー修正履歴と自分のコンパイルエラー修正履歴を比較することが興味深いかどうかを聞いた質問の回答である．被験者の 89%が少なくともやや興味を持っていると答えた．被験者は他者と自分のコンパイルエラー修正の傾向の違いに気が付き，興味を持ったと考えられる．

図 11 は本システムを使うことでコンパイルエラーへの恐怖感が軽減されたかを聞いた質問の回答である．被験者の 44%が少なくともコンパイルエラーへの恐怖感がやや軽減されたと回答した．本システムを使うことで部分的にコンパイルエラーに対する恐怖感を軽減できたと考えられる．

図 12 は想像していたコンパイルエラー修正数と実際の修正状況との差異を聞いた質問の回答である．被験者の 66%がコンパイルエラー発生数が想像よりもやや多いと回答した．被験者は自分が想像しているよりもコンパイルエラーを修正していることに気がついたと考えられる．

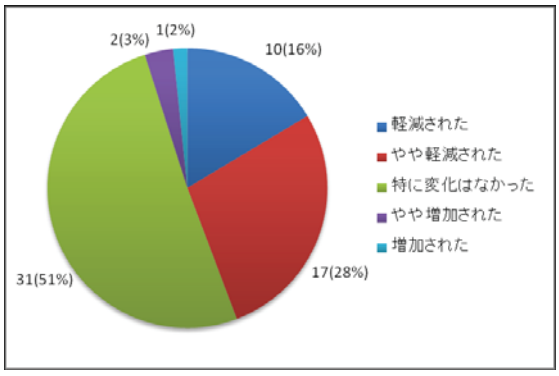

図 11　「この演習を通してコンパイルエラーへの恐怖感は軽減されましたか？」の回答結果

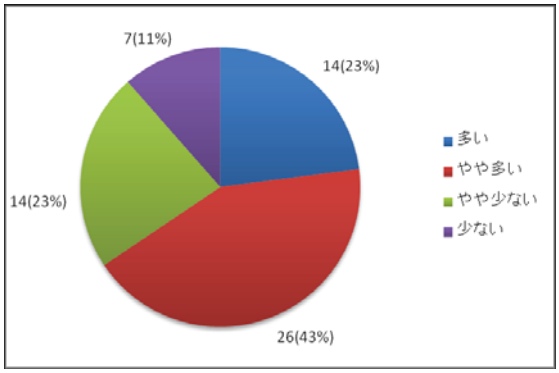

図 12　「自分の想像より，コンパイルエラーの発生数は多いと思いましたか？」の回答結果

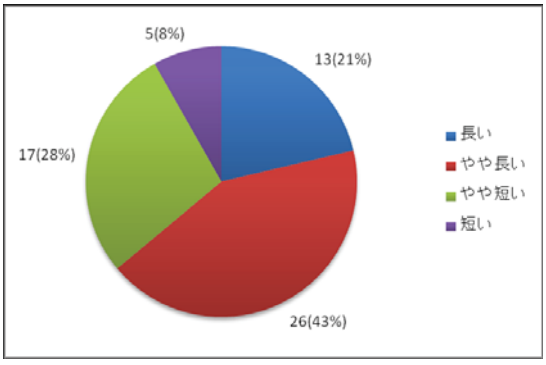

図 13　「自分の想像より，コンパイルエラーの修正時間は長いと思いましたか？」の回答結果

図 13 は想像していたコンパイルエラー修正時間と実際の修正状況との差異を聞いた質問の回答である．被験者の 64%がコンパイルエラー修正時間が想像よりも長いと回答した．被験者は自分が想像しているよりもコンパイルエラーを長く修正していることに気がついたと考えられる．

図 14 は想像していた修正経験のあるコンパイルエラーの種類の数の想像と実際の修正状況の差異を聞いた質問の回答である．被験者の 58%が修正したコンパイルエラーの種類は想像よりもやや少ないと答えた．被験者は修正したコンパイルエラーの種類が少ないので，覚えるべきコンパ

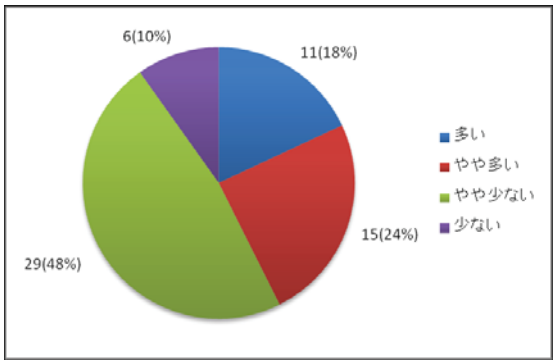

図 14　「自分の想像より，発生したコンパイルエラーの種類は多いと思いましたか？」の回答結果

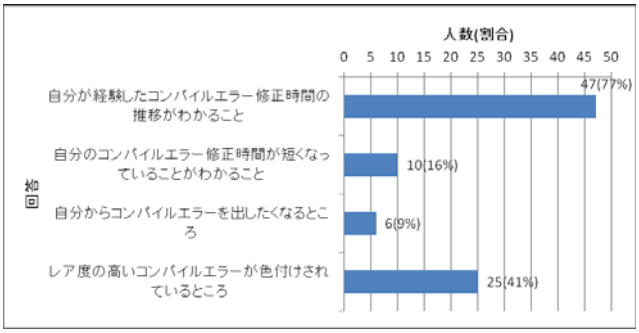

図 15　「修正時間推移グラフ一覧表のどこが面白いと思いますか？」の回答結果 (N=61)

イルエラー修正方法も少ないことに気づいたと考えられる．

図 15 は本システムの長所について聞いた質問の回答である．複数回答可とした．被験者の 77%がコンパイルエラー修正時間の推移がわかることを長所として上げており，こちらが意図した通り被験者がシステムを利用できていることがわかる．被験者の 41%はレア度による修正推移グラフの色付けのデザインを長所としてあげており，このデザインは被験者にとって受け入れやすいものであると考えられる．

図 16 は本システムの短所について聞いた質問の回答である．複数回答可とした．被験者の 52%がコンパイルエラー修正時間を短くする方法がわからないこと，44%がコンパイルエラーが発生した状況がわからないところを挙げている．現状ではコンパイルエラーの発生原因・修正方法を学習する補助が不十分であることがわかる．

図 17 は本システムが学習の役に立つかどうかを問う質問の回答である．被験者の 77%が少なくとも学習のやや役に立ちそうであると答えている．被験者は自らのコンパイルエラー修正情報を分析することが有益であると感じていることがわかる．

表 1 は自由記述欄の回答結果である．

7.　考察

7.1　コンパイルエラーに対する恐怖感の軽減

コンパイルエラーに対する恐怖感が軽減された主な要因

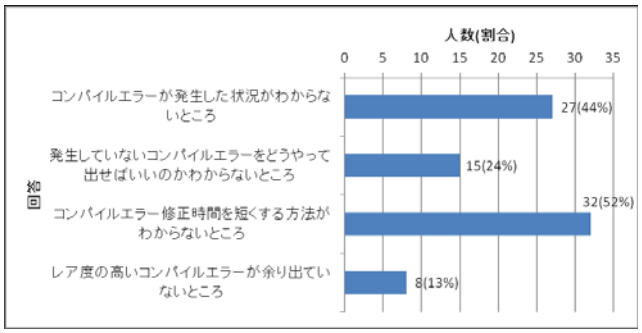

図 16　「修正時間推移グラフ一覧表のどこがつまらないと思いますか？」の回答結果 (N=61)

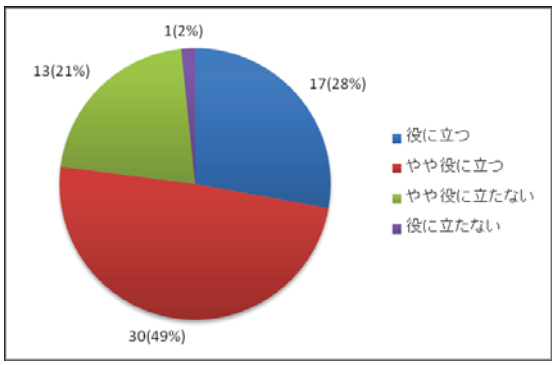

図 17　「修正時間推移グラフ一覧表は学習の役に立つと思いましたか？」の回答結果

表 1　自由記述欄回答

回答者 ID	質問
1	これをひと通りやるだけで，コンパイルエラーに対する恐怖感が激減した．
2	2 回目以降のコンパイルエラー修正時間が短くなっている．
3	当然のことではあるが，修正時間を要する時間が短くなっているのが明らか．
4	右下がりのコンパイルエラーが少なく，まだコンパイルエラーの修正に慣れていないことがよくわかった．
5	自分のグラフは振動型だった．
6	コンパイルエラーの種類によって得意不得意がある．
7	このままでも面白いと感じた．コンパイルエラーについて振り返ることができてよかった．
8	レア度が高いコンパイルエラーをたくさん出していて，羨ましかったです．
9	全然自分のと違うグラフでとても見ていて面白い．

は，本システムを使い，コンパイルエラーに対し漠然と抱いていたイメージと定量的・定性的に示された実際のコンパイルエラー修正の状況との差異を埋めることができたことである．

図 12，図 13，図 14 の回答結果から被験者は以下のことに気がついたと考えられる．

- 修正したコンパイルエラー数が想像よりも多いことから，コンパイルエラー修正の経験は十分である．

- コンパイルエラー修正時間が想像よりも長いことから，コンパイルエラー修正の経験は十分である．
- 修正したコンパイルエラーの種類が少ないので，覚えるべきコンパイルエラー修正方法も少ない．

この内でも特に．被験者は発生したコンパイルエラーの種類が少ないことに気がつくことが重要である．修正したコンパイルエラーの種類が少ないということは，被験者は同じようなコンパイルエラーばかり直していると言える．被験者はよく出るコンパイルエラーを何度も修正しているため，それらの修正方法を身につけることができていると自覚できる．表 1 の ID2,ID3 の意見のようにコンパイルエラー修正方法が身につき，修正時間が短くなっていることについて述べている被験者もいた．

したがって，よく見るコンパイルエラーについては修正方法が身についているという事実に気がつくことにより，コンパイルエラーに対する恐怖感が軽減されたと考えられる．5.1 節の仮説は部分的に支持される．

一方，表 1 の自由記述欄で，ID4，ID5 のように筆者の想定よりもコンパイルエラー修正時間が短くなっていない被験者もいる．コンパイルエラー修正の学習が順調でないと感じている被験者は，コンパイルエラー修正に対して恐怖感が軽減されにくいと考えられる．

7.2 コンパイルエラー修正の学習に対する学習者の興味

図 9 の結果から，被験者の大半がコンパイルエラー修正を分析する作業を興味深く思っていることがわかる．表 1 では，ID6 のように自分のコンパイルエラー修正に対する癖を発見したことや，ID7 のようにシステムを利用することを面白く感じていることが挙がった．

これらの意見の要因として，本システムによってコンパイルエラー修正情報を定量的・定性的に示すことで，被験者が自らのコンパイルエラー修正について考える機会が生まれたことが挙げられる．コンパイルエラーについて考える機会が生まれたことにより，被験者がコンパイルエラー修正について興味を持ったと考えられる．

図 10 の結果から，学習者は他の人のコンパイルエラー修正情報を見ることに興味を持っていることがわかる．表 1 では，ID8 のように修正したコンパイルエラーの種類のレア度に着目した比較を行ったことや，ID9 のように自分とコンパイルエラー修正の状況が全く異なることが挙がった．筆者による観察では，被験者同士でレア度を基準とした積極的な比較を行っており，修正したコンパイルエラーについて話し合っている様子が見られた．

ID8 のように自分が経験したことがないコンパイルエラーを羨ましく思うことは，まだ修正したことがないコンパイルエラーを修正してみたいという意欲に繋がると考えられる．ID9 のように他者と比較することにより，自らのコンパイルエラー修正をより深く考えることができ，学習意欲の刺激に繋がると考えられる．

図 17 の結果から，被験者はコンパイルエラー修正を分析することは有意義であると考えている．学習者は本システムを使うことにより，コンパイルエラーの修正方法・発生原因について理解しようとしていることがわかる．

以上より，本システムを利用することで，学習者のコンパイルエラー修正の学習の動機付けに繋がると考えられる．

8. 結論

学習者が自らのコンパイルエラー修正を観察・分析可能なシステムの開発を行った．文系の大学生約 100 名に対し講義時間内に試用実験を行った．被験者は興味を持ってシステムを利用し，コンパイルエラー修正の学習の動機付けに効果があることが確認された．システムを利用することで学習者のコンパイルエラーに対するイメージと実際の修正状況の差異を埋めることができた．学習者が想像以上にコンパイルエラー修正できていると自覚できたことにより，部分的にコンパイルエラーに対する恐怖感軽減効果が確認できた．

参考文献

[1] 榊原康友, 松澤芳昭, 酒井三四郎：“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”, 研究報告コンピュータと教育 (CE), Vol.2013-CE-118, No.8, pp.1-8, 2013

[2] 山本 耕大, 春原 将寿, 大金 克紀, 中村 勝一, 横山 節雄, 宮寺 庸造：“エラー要因事例ベースの動的学習手法を導入した C 言語教育システムの開発と基礎的評価 (Web 技術の教育利用/一般)”, 電子情報通信学会技術研究報告. ET, 教育工学 108(146), pp.67-72, 2008

[3] 高橋 功欣：“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に関する研究”, 修士論文, 三重大学大学院工学研究科, 2011

[4] 小島 佑介：“プログラミング演習における効率のよい指導のための間違いの早期指摘に関する研究”, 修士論文, 三重大学大学院工学研究科, 2011

[5] Björn Hartmann, Daniel MacDougall, Joel Brandt, Scott R. Klemmer：“What Would Other Programmers Do? Suggesting Solutions to Error Messages”, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp.1019-1028, 2010

[6] 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.2012, No.4, pp.121-128, 2012

[7] 松澤芳昭, 岡田健, 酒井三四郎：“Programming Process Visualizer:プログラミングプロセス学習を可能にするプロセス観察ツールの提案”, 情報処理学会シンポジウム論文集, Vol.2012, No.4, pp.257-264, 2012

[8] CreW Project：“Turtle Cafe -タートルと Java で学ぶ論理思考とプログラミング-
http://crew-lab.sfc.keio.ac.jp/lectures/2009s_tcafe/html4/index.html”