

プログラミング導入教育におけるコンパイルエラー修正時間の分析

榊原 康友[†] 松澤 芳昭[‡] 酒井 三四郎[‡]
 静岡大学大学院情報学研究科[†] 静岡大学情報学部[‡]

1. 研究の背景と目的

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となる。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種類を分析した研究がある^[1]。しかしながら、コンパイルエラーの修正時間を分析した研究はない。そこで本研究では、Java 言語のコンパイルエラーの修正時間に注目し、修正時間の算出アルゴリズムの提案と統計分析を行った。プログラミング教育に活用できるデータを得ることが本研究の目的である。

2. 修正時間算出アルゴリズム

コンパイルエラーの数と修正時間の基礎データは、学習者がプログラム作成時に使用したエディタの操作履歴データより算出する。以下、その際に開発した修正時間の算出アルゴリズムについて述べる。

Java 言語の修正時間算出アルゴリズムは、以下の 7 パターンにまとめられる。図 1 から図 7 において横軸は時間、縦軸はエラーを表現する。黒塗りの図形はエラーの発生時点を表し、白塗りの図形はそのエラーの修正時点を表現する。

2.1 パターン 1

1 つのエラーが次のコンパイル時に修正された場合である。この時の修正時間は $T=t_1$ となる。

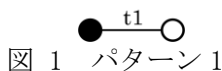


図 1 パターン 1

2.2 パターン 2-A

1 つのエラーを修正しようとした時に、そのエラーが別のエラーメッセージになった場合である。この時、それらのエラーを異なるエラーとして考える。この時の修正時間は $T_A=t_1, T_B=t_2$ となる。

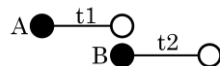


図 2 パターン 2-A

2.3 パターン 2-B

1 つのエラーがその後のコンパイル時に何度か

An Analysis of Compile Error Correction Time in Introductory Programming Education

[†]Kosuke SAKAKIBARA [‡]Yoshiaki MATSUZAWA

[‡]Sanhiro SAKAI

[†]Graduate School of Informatics, Shizuoka University

[‡]Faculty of Informatics, Shizuoka University

修正できない場合である。図 3 のような場合の修正時間は $T=t_1+t_2$ となる。時間 t_n まで修正できない場合は $T=t_1+t_2+\dots+t_n$ となる。

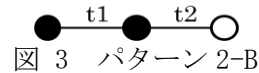


図 3 パターン 2-B

2.4 パターン 3-A

複数のエラーが一度のコンパイルで修正された場合である。この時の修正時間は $T_A=T_B=\dots=T_n=t_1/n$ となる。

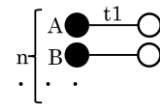


図 4 パターン 3-A

2.5 パターン 3-B

複数のエラーが発生している時に、エラーを 1 つずつ修正した場合である。図 5 のような場合の修正時間は $T_A=t_1, T_B=t_2$ となる。

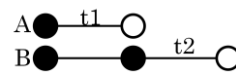


図 5 パターン 3-B

2.6 パターン 4

コンパイル毎にエラーが修正されないままエラー数が増え、それらを複数同時に修正した場合である。

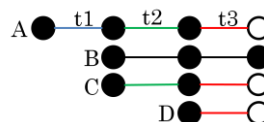


図 6 パターン 4

例えば、図 6 のような場合の修正時間は $T_A=t_1+t_2/2+t_3/3, T_C=t_2/2+t_3/3, T_D=t_3/3$ となる。これは修正されたエラーについて t_1 に関わるエラーが A のみであるため t_1 が T_A に加算、 t_2 に関わるエラーが A, C の 2 つであるため $t_2/2$ がそれぞれに加算、 t_3 に関わるエラーが A, C, D の 3 つであるため $t_3/3$ がそれぞれに加算されている。

2.7 パターン 5

意味解析エラーが発生中に構文解析エラーが発生した場合である。この例を図 7 に示す。

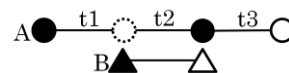


図 7 パターン 5

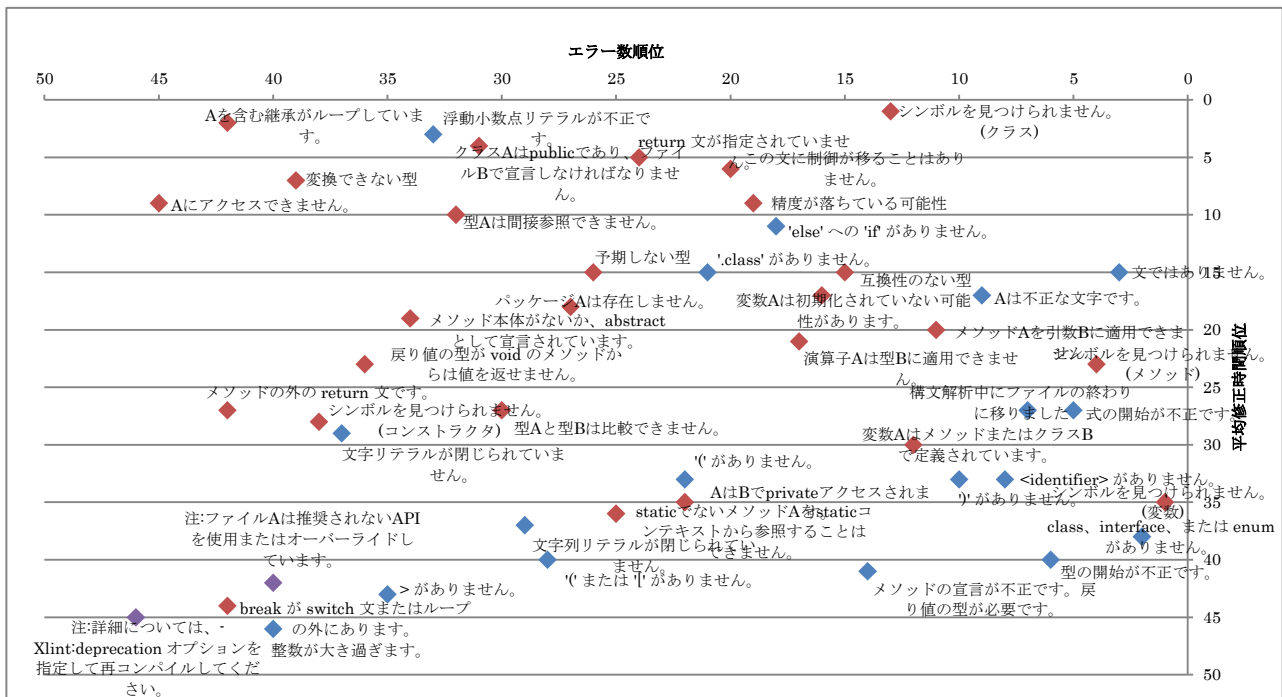


図8 エラー数順位と平均修正時間順位のプロット

図7では丸形が意味解析エラーを示し、三角形が構文解析エラーを示している。Javaコンパイラは構文解析を行った後で意味解析を行うが、構文解析時にエラーを検出すると意味解析を行わない仕様となっている。そのため、意味解析エラーが発生している時に新たに構文解析エラーが発生すると、意味解析エラーは修正されていないにもかかわらず表示されなくなる。こうして「隠蔽」された意味解析エラーを破線丸形で示している。パターン5では意味解析エラーを前半と後半で別のエラーとして考える(AとA')図7のような場合の修正時間は $T_A=t1$, $T_{A'}=t3$, $T_B=t2$ となる。

3. 分析対象データ

エラーデータ取得の対象は文科系の大学1年生約100名である。このエラーデータは学生が講義10回分で課された37個の課題を解く過程で得られたものである。エラーと警告メッセージは全66種類、総数は57,378個であった。

4. 分析結果

エラーの平均修正時間を分析する。この時、エラー数が10未満のエラー(19種類)は除外し、「;がありません。」というエラーについては修正時間を考慮しないこととする。

図8はエラーと警告メッセージ46種類を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である。1位に近いほどエラー数が多く、平均修正時間が長いことを示す。図中の赤の点が意味解析エラー、青の点が構

文解析エラー、紫の点が警告を示す。

意味解析エラーが図8中の上方に分布し、構文解析エラーが図8中の下方に分布しているものが多いことから、構文解析エラーよりも意味解析エラーの方が修正時間が長い傾向にあることが分かる。

図8中の右上に分布するエラーはエラー数が多く目にする機会が多いにもかかわらず修正時間が長いものである。学習者にとってここに分布するエラーの修正が難しいと考えられ、教授者がサポートすべきエラーであると言える。

図8中の左上に分布するエラーはエラー数が少ないが修正時間の長いエラーである。エラーが多くなると修正時間がどのように変化するか分からないため、場合によってはサポートが必要となるエラーもあると言える。

図8中の左下に分布するエラーはエラー数が少ないにもかかわらず修正時間が短いエラーである。これらのエラーは修正するのが簡単であると考えられ、ほとんどのエラーはサポートする必要がないと言える。

図8中の右下に分布するエラーは学習者がエラーを修正することに慣れているエラーと言える。

参考文献

- [1] 森本康彦, 飯島正也, 上野真臣, 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ラエラー分析”, 日本行動計量学会大会発表論文抄録集 33, pp. 266-267, 2005