2ZF-03

静的解析結果と実行履歴を組み合わせた メソッド呼び出し可視化によるコードリーディング支援

鈴木英梨花† 酒井三四郎‡

静岡大学大学院総合科学技術研究科† 静岡大学情報学部‡

1 はじめに

プログラム開発では一からプログラムを書くよりも、既存のプログラムに手を加えていく方法をとることが多い。プログラミング学習時も、与えられたソースコードに対して改変を加える演習を行うことがよくある。しかし、他者の作成したソースコードの改変を行うためには、対象のソースコードを読んで理解する必要があり、理解に時間がかかると本来の目的である改変作業が進まなくなるという問題がある。この問題の原因を大きく分けると「ソースコードの構造を理解していない」ことと「挙動を理解していない」ことの 2 点が考えられる。

上記の問題と原因を解決し、既存のソースコードの理解を容易にするため本研究ではコードリーディングを支援するツールの開発を行う。

2 関連ツールと先行研究

ソースコードの構造や関係を理解するためのツールとして、クラス図を自動生成する Udoc[1]、AmaterasUML[2]、astahUML[3]、コールグラフを自動生成する ispace[4]などがある。また、プログラムの挙動などを動的に可視化する研究[5] [6]も行われている。しかしこれらは可視化した図を理解するのに事前知識が必要になることやソースコード全体を一度に可視化することで注目すべき点が不明確になることなどの問題点がある。

3 メソッド呼び出し可視化プラグイン Umbrella の提案

本研究でははじめに述べた現在の問題と原因を解決し、関連ツールや先行研究に生じている問題点を解消したコードリーディング支援ツール Umbrella の開発を行った。

クラス図やシーケンス図のような UML を利用した可視化の場合、その図の意味を理解するために事前に UML について学ぶ必要がある。そのため、本研究では Java でのプログラミングを学んでいれば理解できるメソッド呼び出しを可視化の対象とした。

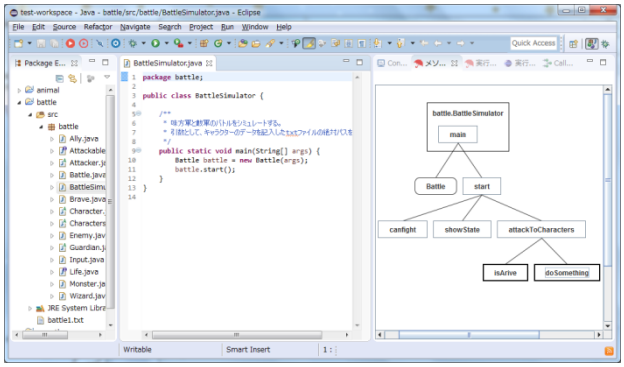

図 1：提案プラグイン使用時の Eclipse の画面

可視化に利用する情報は理解に時間がかかる原因として挙げ点を考慮して決定した。ソースコードの構造は静的解析結果を、挙動は実行履歴を利用すれば理解が容易になると考え、静的解析結果と実行履歴を組み合わせて可視化を行う。静的解析結果を利用し、任意のメソッド呼び出しを 1 段階ずつ可視化する方法と、実行履歴を利用し、メソッド呼び出しを main メソッドから順に 1 段階ずつ可視化する方法の 2 つの方法で実装した。まず、実行履歴による可視化でプログラムの挙動の理解をし、その中でより詳しく確認したいメソッドに対して静的解析による可視化を行うことで、改変を行いたい部分に関連のあるメソッドに絞って読むことができる。

また、プログラム理解の目的はプログラム開発を行うためである。よって理解から開発へスムーズに移行するために統合開発環境 Eclipse のプラグインとして実装した。

図 1 にプラグインを導入した Eclipse の画面を示す。画面左のパッケージエクスプローラーから可視化を実行し、画面右のビューに可視化図が表示される。可視化図内のノードから対応するメソッドのソースコードやそのメソッドが呼び出されている箇所を画面中央のエディタに開くことができるため、ソースコードを確認したり編集したりすることが容易にできる。

4 評価実験

本研究で開発した Umbrella の有効性を評価するために評価実験を行った。本実験では以下の仮説の検証を行う。

仮説「Umbrella の使用によって、プログラムの

Code Reading Support by Method Call Visualization Combining Static Analysis Result and Execution History
†Erika Suzuki
Graduate School of Integrated Science and Technology, Shizuoka University
‡Sanshiro Sakai
Faculty of Informatics, Shizuoka University

理解にかかる時間が短くなる」

4.1 実験方法

静岡大学情報学部情報科学科 3 年の学生 16 人を**表 1** の通り 4 群に分け、問題 A,B の 2 問を出題し、プラグインを使う方法と使わない方法でそれぞれ解いてもらった。問題 A,B はそれぞれ別のソースコードを用意し、ソースコードを読んでもらい、用意した問いに答えてもらうものである。解答中の画面は録画する。実験前にプログラミングと Java についてのアンケートを、問題終了後にもアンケートを実施し、Umbrella の使用時と不使用時を比較して理解の容易さやメリット、デメリットを回答してもらう。

実験結果から仮説の検証を行い、解答結果、アンケート結果、プラグイン使用時の行動を利用し、ツールの有効性を評価する。

4.2 結果と考察

評価実験を行い、平均正答率と平均解答時間を算出した結果は**表 2** の通りである。平均正答率、平均解答時間共にプラグインを未使用時の方がわずかに良い結果となったが、どちらに対しても t 検定を行ったが優位な差は認められなかった。

採点結果、アンケート結果、録画した画面の分析から、理由として被験者の Java の理解度に差があった点やプラグインの効率的な使用方法を実践してもらえかった点、実験に使用したソースコードの規模が適していなかった点などが挙げられる。特に 2 点目については、3 章で述べた「実行履歴による可視化でプログラムの挙動の理解をし、その中でより詳しく確認したいメソッドに対して静的解析による可視化を行う」という使い方の具体的な説明を行わなかったことで、被験者の半数以上が操作の簡単な静的解析による可視化のみを利用していた。

実行ログや録画した画面を分析する中で、16 人中 14 人の被験者がメソッド呼び出しを展開していき、その中の数個のメソッドだけソースコードを開くという使い方を 1 回以上していることが分かった。これは、メソッド名や呼び出し関係を見て、特定の動作をするのに重要なメソッドや編集を行うべきと考えられるメソッドのソースコードを確認していると考えられる。この使い方により、ソースコードをすべて確認するのではなく、必要な部分のソースコードのみを確認するという効率的なコードリーディングができる。このような使い方をしやすくするために、課題として操作手順を減らすなどユーザビリティの改善が挙げられる。

実験後のアンケートでは「どちらの方法のときの方が問題を解きやすいと思いましたか？」という質問に対して 16 人中 15 人がプラグインを使用したときと回答した。その理由として「構造が直感的に理解できたから」など、ソースコードのみを使って読むよりも理解しやすいという意見が多かった。

以上から、本実験で仮説を立証することはできなかったが、操作手順の簡略化や実験の見直しをすることでプラグインの有効性を高めることができるといえる。

表 1：問題とプラグイン使用有無の組み合わせ

群	1 問目	プラグイン	2 問目	プラグイン
1	A	無	B	有
2	B	無	A	有
3	A	有	B	無
4	B	有	A	無

表 2：実験結果

	平均正答率	平均解答時間
プラグイン有	62.1	28 分 56 秒
プラグイン無	67.5	27 分 36 秒

5 おわりに

本研究ではソースコードリーディング支援のためメソッド呼び出しを可視化するプラグインの提案を行った。プラグイン使用によってプログラムの理解は容易になるが、効果は使用者のプログラミングレベルやソースコードに左右されてしまう。また、ユーザビリティの見直しが今後の課題と分かった。

参考文献

[1] AmaterasUMLProject Amateras, http://amateras.osdn.jp/cgi-bin/fswiki/wiki.cgi?page=AmaterasUML (2018/12/19 参照)
[2] Astah, http://astah.change-vision.com/ja/ (2018/12/19 参照)
[3] The UDoc project, http://udoc.sourceforge.net/main/index.html (2018/12/19 参照)
[4] ispace:Ispace-Home browse, http://web.archive.org/web/20111203070026/,http://ispace.stribor.de/index.php?n=Ispace.Home (2018/12/19 参照)
[5] 伊藤芳朗,渡邊結,石尾隆,井上克郎: オブジェクトの動的支配関係解析を用いたシーケンス図の縮約手法の提案,情報処理学会研究報告,Vol.2008-SE-160,pp.9-16,2008
[6] 中原進,紫合治:オブジェクト指向プログラムの動作の可視化,情報処理学会研究報告,Vol.2010-SE-167 No.9,pp.1-8,2010