GUIを持つプログラムの理解支援のための可視化システム

佐藤竜也[†]　志築文太郎[†]　田中二郎[†]

Visualization for supporting extension of GUI programs

Tatsuya Sato,[†] Buntarou Shizuki[†] and Jiro Tanaka[†]

1. はじめに

他人の書いたプログラムに機能を追加する場合、開発者はそのプログラムを理解するために、実行の外的振る舞いや関数呼び出し、クラス階層などの要素を把握する必要がある。ソースコードのみからこの把握を行う場合、プログラムの規模が大きくなるとプログラムの理解に必要な部分を即座に見出すことは難しい。

また、現在開発されるソフトウェアの多くはGUI(Graphical User Interface) を持つ。GUI はマウスやキーボードを用いた操作に応じて処理を行い視覚的なフィードバックを返す特徴を持つ。そこで本研究ではこの特徴を活かしたプログラム可視化手法を提案することでGUI を持つプログラムの理解を支援し、機能追加時にソースコードを読む作業のコストを最小化し作業効率を向上させることを目的とする。既存の研究として[1)~4)] 等があるが、我々は以下のような可視化を行う。

可視化する要素

GUI を持つプログラムに機能追加を行う場合のプログラムの理解すべき重要な要素は以下である。

(1)　各操作のプログラム全体に対する実装部分
(2)　関数呼び出し階層
(3)　クラス階層

本研究ではこれらの要素を効果的に可視化する。

インタラクティブ性

プログラムへの操作と可視化が別々に行われることによる時間差は、開発者がいつ、どのような操作を行ったかを忘れさせるので、操作と可視化情報を結び付けにくい。これに対して、我々はGUI 実行画面の操作に対してインタラクティブに反応する可視化システムを構築する。

† 筑波大学コンピュータサイエンス専攻
Department of Computer Science, University of Tsukuba

2. システム概要

基本表示

図1 にシステムの概観を示す。上部はコントロール部、中央はグラフ表示部、下部はサムネイル表示部である。プログラム理解にはソースコードを見る必要があるため、本システムではソースコードそのものを表示に利用する。ソースコードはクラスごとにクラス名付きパネルとしてグラフ表示部に縮小表示される。各パネルは親子関係で結び付けられて木構造状に配置される (図1、親子関係ツリー)。

コントロール部にはボタンが配置されていて、開発者は対象プログラムの起動、停止、ステップ実行等を行える。さらに開発者は起動した対象プログラムのGUI を操作することができる。システムはGUI への操作に反応して、インタラクティブに表示を更新する。

システムは随時、操作によって実行されたソースコード行を強調表示し関数呼び出しを矢印で結び順序付けを行う (図1、プログラム実行部)。操作によるプログラム実行が終了すると、その操作に対するグラフ表示が完成し、それ以降静的に閲覧することができる。このように遠目から眺めるようにプログラムの実行を見ることで、開発者は各操作のプログラム全体に対する実装部分を知ることができる。

サムネイル表示部には、操作前と後での対象プログラムのGUI の画面遷移がサムネイルで表示される (図1 参照)。このサムネイルは選択することが可能で、左の画面を選択すると前、右の画面を選択すると後の操作の実行の静的グラフを閲覧することができる。

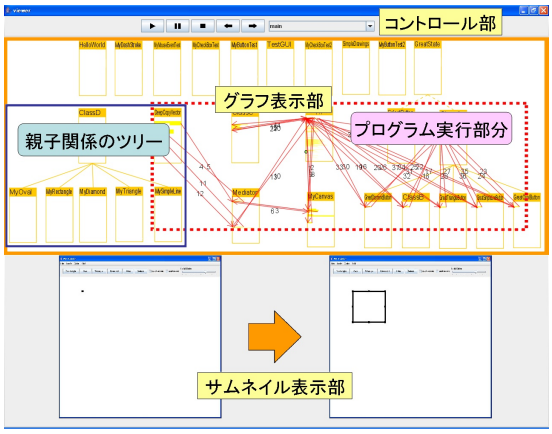

図 1 システムの概観

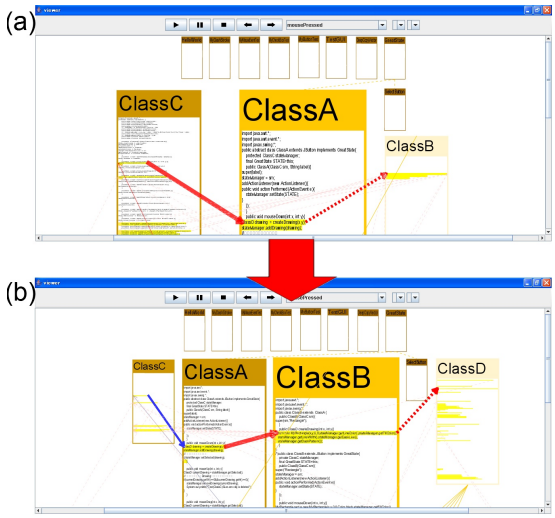

図 2 ズーミング機能を用いた詳細表示

詳細表示

操作ごとのプログラム実行を詳細に把握するには、静的グラフを構成するエッジを順に追えばよい。そこで我々は静的グラフの注目している部分のソースコードが見えるように拡大しながら、エッジを順に追う詳細表示を実装した。注目部分は見ている時点の実行クラスとその前後のクラス間の関数呼び出しであると考え、それらのクラスのパネルと呼び出しのエッジを拡大表示する。またそれ以外の呼び出しは注目していない情報とみなし、パネルを暗色で縮小表示する。エッジに関しても透過処理を施して目立たなくさせる。エッジはその時点の以前と以後の呼び出しであるかがわかるようにそれぞれ別の色で描画される。このように開発者は注目部分だけを取り出しながら、プログラムの実行の様子を追うことが可能となる。

ズーミング機能を用いた詳細表示を図 2 に示す。このように詳細表示ではある時点での注目部分だけがズームして表示される。図 2a はある時点の詳細表示である。今 ClassA が実行されていて、次に ClassB の実行が控えている。開発者がボタンを押してステップ実行を行うと図 2a から図 2b のように表示が変化する。図 2a では ClassA から ClassB への呼び出しのエッジは次の時点での呼び出しなので点線表示されているが、次のステップである図 2b では ClassB が実行されているためエッジは実線表示されている。

3. 実　　装

本研究では Java を対象言語にした。Java プログラムの動作を解析するためにはソースコードとクラスファイルから、メソッドやフィールドといったクラス構成、およびプログラム実行中のメソッド呼び出しやフィールドの変化といった動的情報を取得する必要がある。特に動的情報はトレースとして実行時に取得する。動的情報を取得するために JavaVM の実行を明示的に制御し、各クラスの情報を観察することができる JDI(Java Debug Interface)API を使用した。システムは、まずソースコードから静的なクラス構成を取得してシステムの概観を表示する。その後、JDI を用いて対象となる GUI を持つプログラムを立ち上げる。開発者がプログラムを操作することによりイベントが発生すると、JDI によってその場で動的にプログラムの実行を解析し、実行の情報を表示に反映させる。表示のズーミング機能の実装には Piccolo.Java 1.2 を利用した。Piccolo はズーミングインタフェースの構築をサポートするツールキットである[5]。

参　考　文　献

1) Stephen G. Eick and Joseph L. Steffen and E. Summer : Seesoft - A Tool For Visualizing Line Oriented Software Statics, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol.18, No.11, pp.957–968 (1992).
2) 寺田実：電子紙芝居 Electoronic Paper-Slide Show，日本ソフトウェア科学会 WISS'98，pp. 181–186 (1998).
3) 久永賢司, 柴山悦哉, 高橋伸：GUI プログラムの理解を支援するツールの構築，日本ソフトウエア科学会第 17 回 (2000 年度) 大会 (2000).
4) Storey, M.-A., C. Best, and J. Michaud : SHriMP views - an environment for exploring Java programs, IEEE International Workshop on Program Comprehension '2001 (2001).
5) Piccolo: http://www.cs.umd.edu/hcil/piccolo/.