

科学研究費助成事業 研究成果報告書

平成 26 年 6 月 13 日現在

機関番号：14603

研究種目：挑戦的萌芽研究

研究期間：2011～2013

課題番号：23650016

研究課題名(和文) データフロー検索と可視化によるオブジェクト指向プログラム理解の支援手法

研究課題名(英文) A method that supports understanding of object-oriented programs by querying and visualizing dataflows

研究代表者

久米 出 (Kume, Izuru)

奈良先端科学技術大学院大学・情報科学研究科・助手

研究者番号：10301285

交付決定額(研究期間全体)：(直接経費) 2,700,000円、(間接経費) 810,000円

研究成果の概要(和文)：オブジェクト指向フレームワークを利用した開発手法の普及と共に、ソフトウェア保守の観点からフレームワークアプリケーションの理解が重要な課題として挙げられるようになった。本研究では特にフレームワークの誤用によって発生する障害の解決を目的とした動的解析手法を提案する。本提案手法では誤用特定を支援するために、誤用を含むコードの実行から障害発生に至る、所謂感染の連鎖の発生を示唆する部分をトレース(プログラムの実行履歴)中の依存関係から抽出する。研究代表者は本手法の支援を実現するために感染の連鎖候補を可視化するツールを試作し、本格的な評価実験に向けた予備的な調査を実施した。

研究成果の概要(英文)：In the wide spread use of object-oriented frameworks in software development, program understanding of framework applications becomes an important topic of software maintenance. We propose a dynamic analysis method that aims to help maintainers find framework misuses, which cause failures. Our method supports maintainers by extracting candidates of a so-called chain of infection, which starts by executing a program code containing a framework misuse and results in a failure, from an execution trace. We developed a prototype tool that visualizes candidates of a chain of infection. We conducted a preliminary evaluation for the preparation of an evaluation experiment in future.

研究分野：総合領域

科研費の分科・細目：情報学・ソフトウェア

キーワード：ソフトウェア工学 デバッグ支援 動的解析 オブジェクト指向プログラミング アプリケーションフレームワーク プログラム理解

1. 研究開始当初の背景

ソフトウェア開発に於けるオブジェクト指向プログラミングの普及と共に、そのプログラム理解の支援手法が求められるようになってきている。オブジェクト指向プログラムを理解するためには、プログラムの実行時に登場するオブジェクトやそのクラスが相互のやりとりの中で果たす責務を特定する必要がある。これはシステムの開発者の「非局在化された意図(Delocalization Plans)」に従って決定される。文書として残されていない非局在化された意図を復元する事は極めて困難な問題であるとされている。

非局在化された意図の復元が困難である第一の理由は、通常利点として挙げられるオブジェクト指向プログラミングに固有な技法にある。例えばオブジェクト指向プログラミングに於ける多相型はプログラムの再利用性を高める一方で、実行時に参照されるオブジェクトのクラスやメソッドをソースコードから特定する事を困難にする。またクラス継承の階層が深くなる事によってソースコードの読解時に多数のクラス階層を行き来する、所謂 YO-YO 問題が発生する事が珍しくない。

非局在化された意図の復元を困難とするもう一つの理由はオブジェクトの状態に関する副作用の存在である。対話的なオブジェクト指向プログラムでは利用者の操作によってシステム内部の状態が変更され、変更された状態に依存する形で以降の操作時のシステム挙動が決定される。またフレームワーク上に構築されたアプリケーションでは、しばしばフレームワーク内部の状態を正しく変更するメソッドの呼び出しが必要とされている。こうした状態の変更と挙動の関係を特定するためには、メソッドの呼び出し関係を追うだけでなく、データの流れの解析も必要である。

我々の知る限り現在に至るまでこの難問の解決に正面から取り組んだ研究は存在しない。研究代表者はこうした現状を踏まえ、データフローとメソッド呼び出しの双方の視点から非局在化された意図の復元を支援する手法の開発を開始した。

2. 研究の目的

本研究の最終目標は前述した非局在化された意図の復元の問題の解決である。目標達成に向けて、メソッド呼び出しとデータフロー間の関係に関して開発者(保守作業員)による理解を支援する手法を開発する。実用的なプログラムの実行時に於けるメソッド呼び出しやデータの流れは極めて複雑であるため、手法の利用者(開発者や保守作業員)に有用な

内容を示す抽象化・可視化の実現を目指した。また利用者にとって必要な情報を取得するための検索に関する考察も行った。

一般にプログラムの挙動の抽象化にはその基盤となるモデルの定義が必要となる。モデルを導入する目的は「非局在化された意図の復元」であるため、モデルの妥当性を評価するためには以下の二点の検討が必要であった：

- 非局在化された意図とは具体的に何を意味するのか
- 非局在化された意図の復元をどう確認するのか

一般に非局在化された意図は非常に一般的な概念であるため、本研究では「アプリケーションフレームワークの利用規則」に焦点を絞る事にした。現在ソフトウェアシステムの多くがフレームワークのアプリケーションとして開発されている。その中でフレームワークの利用規則の違反による不具合が重大な問題として取り上げられるようになってきている。

フレームワークの利用者(アプリケーション開発者)の視点からは、その利用規則は非局在化されている「フレームワーク開発者の」意図に他ならない。アプリケーション開発者によるフレームワークの誤用はこのフレームワーク開発者の意図の理解の誤りを意味している。よって、誤用箇所を特定・修正した時点でその意図が復元されたと見做す事が出来る。以上の考察より研究の適用範囲を限定し過ぎる事なくその焦点を絞る事が可能であると考えられる。

上記方針の下に研究代表者らはフレームワークアプリケーション上の誤用箇所の特定を支援する手法を提案した。本手法はフレームワークアプリケーションの実行をフレームワークとアプリケーション固有部分の相互のやりとりとしてモデル化する。またアプリケーション固有部分の実行内容とフレームワーク内部の状態変化の対応付をモデル上で表現する事によって「フレームワーク開発者の意図に反した」アプリケーション固有コードの特定を支援出来ると考えた。

3. 研究の方法

本研究はフレームワークの誤用の特定支援を目的としている。フレームワークの誤用は(1)アプリケーション固有部分のコードがフレームワーク開発者の想定していない処理を実行し、(2)それによってフレームワーク内部で異常な状態が発生し、(3)以降の実行で以上な状態の変遷が実現する、所謂感染の連鎖が起こり、(4)最終的に障害が発生して

その存在が明らかになる。

誤用を解決するためには上記の感染の連鎖の発端となったアプリケーション固有部分のコードとその実行箇所を特定する必要がある。本来であればフレームワーク開発者がこうした誤用を防ぐために利用に関する規則を文書として明示化すべきである。しかしながら現実の多くのフレームワークにはこうした文書が整備されておらず、アプリケーション開発者はフレームワーク開発者の想定を推測しながらアプリケーション固有なコードを開発せざるを得ない。

こうした状況の中でアプリケーション開発者は感染経路を特定する事によってその発端となる誤用を特定せざるを得ない。しかしながら上記 [1. 研究開始当初の背景] で述べたオブジェクト指向的なプログラミング技法の存在のために、ソースコードの記述から感染経路を特定する事は極めて困難である。この種の問題を回避する常道としてしばしば動的解析が導入される。一般に動的解析はプログラムコードをその本来の機能を損なわない形で拡張し、拡張コードが実行時に生成したトレース(実行履歴)から有用な情報を抽出する技術である。研究代表者は誤用による感染経路を特定する動的解析手法を開発する事にした。ここで解析対象言語として Java を選定した。

研究代表者は動的解析手法の開発に当たって誤用から障害に至る感染の連鎖過程に関する考察を行った。アプリケーションの実行時にはフレームワークからアプリケーション固有な拡張を実装するメソッドが呼び出される。逆にフレームワークの内部状態を変更するメソッドがアプリケーション固有部分からしばしば呼び出される。このようにアプリケーションの機能はフレームワーク側とアプリケーション固有側の相互のメソッド呼び出しによって実現されている。誤用に起因する感染の連鎖を特定するためには、こうした相互の呼び出し関係の中で形成される制御や依存関係の解析が必要である。

こうした依存関係の解析によって感染の連鎖の発生場所がある程度絞り込まれる。しかしながら通常こうした解析結果は人間が理解するにはあまりにも複雑であるため、効率的な誤用の特定に用いる事は困難である。また、アプリケーション開発者は通常フレームワークの実装の知識を有していないため、フレームワーク側のコードの実行を内容から感染の連鎖を特定する事は、例え依存関係が解析されていたとしても相当に困難である。

上記の問題分析に基づいて、研究代表者は (1)依存関係から感染の連鎖が発生している可能性のある部分を抽出し、(2)抽出部分に

於ける実際の感染の連鎖の有無の判断を支援するための抽象化を行う動的解析手法を提案した。

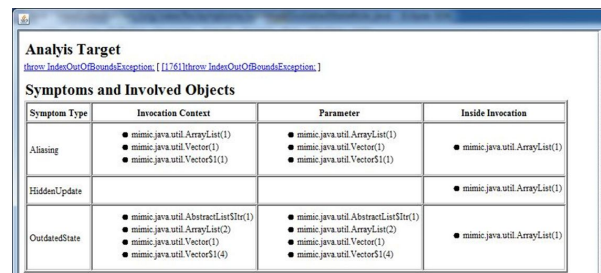
提案手法では、感染の連鎖の可能性を特定するために、フレームワーク側から呼び出されたアプリケーション固有なメソッドが「呼び出し側に隠れて」フレームワークの内部状態を変更している実行箇所を外形的に特定するアルゴリズムを考察した。さらにフレームワーク側がその内部状態の変更を「知らずに利用している」実行箇所も外形的に特定するようにした。こうした実行箇所は感染の連鎖の「兆候」と名付けた。

依存関係の解析で得られた結果のうち、特定された兆候に経由する部分を感染の連鎖が発生している可能性のある部分として抽出するようにした。こうした抽出部分(関連の連鎖候補)は兆候と関連付けられる形でモデル化される。

デバッグの支援の一環としてトレースに対する様々な検索機能の研究が実施されている。しかしながらこうした検索手法では作業者がプログラムの実装の詳細を熟知している事が前提となっている。本研究の想定では作業(アプリケーション開発者)が検索に必要な知識(フレームワークの実装の知識)を有していないため、作業による検索の代わりに兆候の特定機能を実装するようにした。

4. 研究成果

上記 [3. 研究の方法] で提案した手法を解析ツールとして試作した。試作ツールでは抽出された「兆候」に關与するオブジェクトのクラスの一覧を表として図示する機能と兆候を障害との依存関係を可視化して表示する機能を実装した。第三者が開発した実用的なフレームワークアプリケーションの誤用事例に対して試作ツールを適用した。兆候に關与するオブジェクトのクラスの表を図 1 に、兆候の依存関係の可視化を図 2 に示す。



The screenshot shows a window titled "Analysis Target" with a URL. Below it is a table titled "Symptoms and Involved Objects". The table has four columns: Symptom Type, Invocation Context, Parameter, and Inside Invocation. The rows are: Aliasing, HiddenUpdate, and OutdatedState. Each row lists specific Java classes and methods involved in the symptom.

Symptom Type	Invocation Context	Parameter	Inside Invocation
Aliasing	<ul style="list-style-type: none">mimic.java.util.ArrayList(1)mimic.java.util.Vector(1)	<ul style="list-style-type: none">mimic.java.util.ArrayList(1)mimic.java.util.Vector(1)	<ul style="list-style-type: none">mimic.java.util.ArrayList(1)
HiddenUpdate			<ul style="list-style-type: none">mimic.java.util.ArrayList(1)
OutdatedState	<ul style="list-style-type: none">mimic.java.util.AbstractList\$Iter(1)mimic.java.util.ArrayList(2)mimic.java.util.Vector(1)mimic.java.util.Vector\$1(4)	<ul style="list-style-type: none">mimic.java.util.AbstractList\$Iter(1)mimic.java.util.ArrayList(2)mimic.java.util.Vector(1)mimic.java.util.Vector\$1(4)	<ul style="list-style-type: none">mimic.java.util.ArrayList(1)

図 1 兆候に關与するオブジェクトのクラス

予備的な調査としてソフトウェア開発企業の技術者に試作ツールの可視化結果を示し

て定性的な評価を依頼し、以下の結果を得た：

- フレームワークの誤用はしばしば現実に発生し、多くのソフトウェア開発を停滞させる深刻な問題である。
- 副作用によって障害が引き起こされる関係の可視化は原因の特定する上で有用である。
- 図 1 の表中のクラスとプログラム中の他のクラスの関連を示す情報が欲しい。
- 副作用が実際にフレームワーク側にとって予期せぬものか否かを判定する事が困難である事が多いため、その判定を支援する機能が必要である。
- 上記に関連して本来あるべき処理過程を抽象的に表現し、兆候の導入によってあるべき処理から逸脱した仕組みを明らかにする機能が必要である。
- Java のみでなく JavaScript プログラムでもこうした誤用特定が必要となる事例が多いと思われる。
- 実用上の観点から試作ツールの処理速度に関してはさらに改善される事が望ましい。
- オブジェクトの大量生成が処理速度を低下させる主要原因である可能性が高い。



図 2 依存関係に基づく兆候の配置

予備調査の結果、評価実験の実現には支援のための機能の追加と処理速度の向上が必要である事が判明した。これらの改善点を反映させるために試作ツールの改善を実施中である。

本試作ツールが生成・処理するトレースには依存関係を含む様々な種類の実行情報が含まれている。結果としてトレース処理過程で十万個以上のオブジェクトが生成されている。上で述べたようにこうした大量のオブジェクト生成が処理速度の低下の主要原因と考えられる。

一方でこうした依存関係を含む包括的な形で実行内容を記録したトレースを、実用的なプログラムのデバッグに利用した事例は研究代表者の知る限りでは他に存在しない。既

存の動的解析手法では、トレースが大規模化しないよう、解析に必要な実行の特定の側面のみをトレースに記録させている。本試作ツールは市販の PC の上で稼動している。特別なハードウェアを利用する事なく、実用的なプログラムの包括的な実行内容を記録したトレースの処理が可能である事を実証した事になる。この実証も本研究の成果として挙げられる。

5. 主な発表論文等

(研究代表者、研究分担者及び連携研究者には下線)

[雑誌論文](計 0 件)

[学会発表](計 15 件)

[1] Naoya Nitta, Izuru Kume, Yasuhiro Takemura: Identifying Mandatory Code for Framework Use via a Single Application Trace, European Conference on Object-Oriented Programming (ECOOP), 2014 年 7 月, accepted to appear.

[2] Izuru Kume, Masahide Nakamura, Naoya Nitta, Etsuya Shibayama: Toward A Dynamic Analysis Technique to Locate Framework Misuses That Cause Unexpected Side Effects, 15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2014), accepted to appear.

[3] Izuru Kume, Naoya Nitta, Masahide Nakamura, Etsuya Shibayama: A Dynamic Analysis Technique to Extract Symptoms That Suggest Side Effects in Framework Applications, ACM Symposium on Applied Computing (SAC) pp.1176-1178 2014 年 3 月 27 日, Gyeongju, Korea.

[4] 久米出, 新田直也、中村匡秀、柴山悦哉: フレームワーク・アプリケーションに於ける予期せぬ副作用の効率的なデバッグに向けて、ソフトウェアサイエンス研究会 (SS) 2014 年 3 月 11 日, 沖縄県那覇市.

[5] 久米出、中村匡秀、新田直也、柴山悦哉: フレームワーク誤用による副作用の可視化手法、ウィンターワークショップ 2014・イン・大洗 2014 年 1 月 23,24 日, 茨城県茨城郡大洗町.

[6] Naoya Nitta, Izuru Kume and Yasuhiro Takemura: A Method for Early Detection of Mismatches between Framework Architecture and Execution Scenarios, Asia-Pacific Software Engineering Conference (APSEC)

pp.517-522 2013 年 12 月 5 日, Bangkok, Thailand.

[7] 久米出、新田直也、中村匡秀、柴山悦哉: フレームワークアプリケーションに於ける副作用の兆候を抽出する動的解析手法, 第 182 回ソフトウェア工学研究発表会 2013 年 10 月 24 日, 石川県金沢市.

[8] Izuru Kume, Masahide Nakamura, Naoya Nitta, and Etsuya Shibayama: A Feature Model of Framework Applications, 14th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD 2013), pp.511-516, 2013 年 7 月 1 日, Honolulu, USA.

[9] 久米出、中村匡秀、新田直也、柴山悦哉: フレームワークアプリケーションの抽象化のための動的解析手法, 第 179 回ソフトウェア工学研究発表会 2013 年 3 月 11 日, 東京都千代田区.

[10] 久米出、中村匡秀、新田直也、柴山悦哉: 動的解析によるフレームワーク学習に向けて:, ウィンターワークショップ 2013・イン・那須 2013 年 1 月 24,25 日, 栃木県那須郡那須町.

[11] Izuru Kume, Masahide Nakamura, Etsuya Shibayama: TOWARD UNDERSTANDING SIDE EFFECTS IN FRAMEWORK APPLICATIONS, International Conference on Software Technology and Engineering 2012 年 9 月 2 日, Phuket, Thailand.

[12] 久米出、中村匡秀、柴山悦哉: フレームワークアプリケーションの副作用の特徴付け手法, 第 177 回情報処理学会ソフトウェア工学研究会, 2012 年 7 月 20 日, 大阪大学.

[13] 久米出、柴山悦哉: トレース解析手法を利用した逸脱コードの特定, 第 175 回情報処理学会ソフトウェア工学研究会, 2012 年 3 月 15 日, 東京都千代田区.

[14] Izuru Kume, Masahide Nakamura, and Etsuya Shibayama: Toward Comprehension of Side Effects in Framework Applications as Feature Interactions, The 19th Asia-Pacific Software Engineering Conference (APSEC 2012) pp.713-716, 2012 年 12 月 7 日, Hong Kong.

[15] 久米出、柴山悦哉: イベント駆動プログラム理解のための動的解析手法, ウィンターワークショップ 2012・イン・琵琶湖 2012 年 1 月 19,20 日, 滋賀県彦根市.

〔図書〕(計 0 件)

〔産業財産権〕
出願状況 (計 0 件)

名称 :
発明者 :
権利者 :
種類 :
番号 :
出願年月日 :
国内外の別 :

取得状況 (計 0 件)

名称 :
発明者 :
権利者 :
種類 :
番号 :
取得年月日 :
国内外の別 :

〔その他〕
ホームページ等

6. 研究組織

(1) 研究代表者

久米 出 (Kume Izuru)

奈良先端科学技術大学院大学・情報科学研究科・助手

研究者番号 : 10301285

(2) 研究分担者

()

研究者番号 :

(3) 連携研究者

()

研究者番号 :