科学研究費助成事業　　研究成果報告書

平成　２６　年　　６　月　　６　日現在

機関番号：　３４３１５
研究種目：　若手研究(B)
研究期間：　2012～2013
課題番号：　２４７０００３４

研究課題名（和文）開発者の操作履歴に基づきプログラムの理解と変更を支援する統合開発環境

研究課題名（英文）An integrated development environment for supporting program comprehension and change based on developers' operations

研究代表者
大森　隆行（Omori, Takayuki）

立命館大学・情報理工学部・助教

研究者番号：９０５３２９０３

交付決定額（研究期間全体）：（直接経費）　　1,700,000円、（間接経費）　　510,000円

研究成果の概要（和文）：本研究では、ソフトウェアの保守工程における品質低下を防ぐため、開発者が統合開発環境上で行った操作の履歴を利用した開発支援手法について研究を行った。各操作が行われた時点のソースコードを容易に解析できるツールの実装、過去のコード補完操作に着目した補完候補推薦手法の実現、編集操作履歴をより良い粒度に再構成する手法の実現、開発タスクごとの操作履歴の特徴に基づくタスク情報生成等に取り組んだ。

研究成果の概要（英文）：In these two years, we promoted our research work regarding software development support to avoid deterioration of software quality, especially in maintenance phase. Our work is based on developers' past operations performed on integrated development environments. We worked on a tool to easily parse past source code which is restored by using recorded edit operations, a technique that improves an existing code completion mechanism based on past code completion operations, methods for reorganizing recorded operations to improve their understandability, and a tool that generates abstract task information that is suitable for understanding past development work.

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　ソフトウェア開発支援　統合開発環境　ソフトウェア理解　ソフトウェア進化

１．研究開始当初の背景

社会において広くソフトウェアが使用される現状において、ソフトウェアに欠陥がないことは重要である。ソフトウェアの欠陥は、保守作業において既存のプログラムを作り変える際に埋め込まれることが多い。その主な原因として挙げられるのが、既存のプログラムを十分に理解しないまま変更を行うことである。例えば、ソースコード上に同時に変更しなければならない箇所が互いに離れた場所に散在するとき、プログラムの理解や変更に関して適切な支援がなければ開発者が変更を忘れる可能性が高い。このような場合、ソフトウェアに予期しない欠陥が埋め込まれることになる。

このような問題を解決するため、申請者らは、これまでに、ソフトウェア開発において開発者が統合開発環境(IDE)で行った編集操作をすべて記録し、その履歴から有用な情報を抽出する手法について、研究を進めてきた。その成果として、操作履歴記録ツール OperationRecorder と、記録された操作を再生し、効率的に分析するためのツール OperationReplayer を実現した。これらのツールは、類似ツールよりも細かい単位で Java 言語の操作履歴を扱うことが可能であり、過去のソースコードの任意の状態を復元可能であるという特徴を持っている。また、OperationReplayer は、ユーザの利用目的に応じて操作履歴を柔軟に視覚化するタイムラインバーを備えており、操作の時系列分布や種類を短時間で把握することを可能としている。

従来、ソフトウェア進化の研究においては版管理システムから得られるソースコードの差分情報を利用してプログラム変更を推測していたが、本研究のような操作履歴記録手法によって、より詳細に変更を追跡することが可能となった。このことから、操作履歴を利用することで、これまでに実現されてきた変更情報を利用する手法の性能を改善できると期待されている。実際に、操作履歴により、プログラム理解の正確さや早さが改善する例が報告されている。さらに、IDE 上でこのような開発プロセスのデータを利用した支援を行うことの重要性が先行研究において指摘されている。一方、ソフトウェア開発において記録される操作履歴は膨大であるため、広範囲にわたって操作内容を理解したり、目的の操作情報を探し出すことは、いまだ困難である。ソフトウェア保守作業において、保守者が過去のプログラム変更を十分に理解し、誤ったソフトウェア修正を行わないようにするためには、さらなる理解支援、変更支援が必須である。

２．研究の目的

本課題開始当初の研究目的を述べる。

(1) 理解しやすいプログラム変更情報の生成

保守者が過去のプログラム変更情報を見たときに、すばやく正確にその内容を理解するためには、どのような変更履歴情報の構成が適切か、どのような情報が有用かを明らかにする。具体的には、履歴中の雑音(役に立たない情報)の除去や最適な操作の切れ目(理解単位)の設定をどのように行うべきか、プログラムの構文・意味や開発者の作業の種類が理解に有効かを調査する。

(2) 様々な観点に基づく操作履歴の特徴

様々な観点(例えば、特定の開発者やタスクの種類)に着目して操作履歴を見たときに操作履歴にどのような特徴が表れるかを分析する。例えば、操作の対象が同じ箇所でも、該当箇所の理解を行う場合とデバッグを行う場合では、閲覧や変更の傾向に違いが生じる。本研究では、特定の成果物間の強い共起関係や時系列の出現パターンをマイニングすることで、後述する変更推薦手法の実現を目指す。

(3) 過去の操作に基づく操作の推薦

(2)で明らかにした操作履歴の特徴に基づき、開発者が次に閲覧すべき箇所、変更すべき箇所の候補を推薦する手法を考案する。これをツールとして実装し、その推測精度を明らかにする。

３．研究の方法

ここでは、研究方法の具体的内容について述べる。

(1) 操作履歴の再構成手法

OperationRecorder により得られる操作履歴は、開発者の実際の入力操作に応じて、操作の構造が変化する。この結果、入力ミス等により操作が意図せず分断されたり、IDE の入力支援機能による自動操作によって直感的でない操作が生成されるという問題がある。そこで、操作の理解性を高めるため、記録された操作履歴を再構成する手法を考案した。操作の基本的な処理(分解・連結・順序変更等)を応用し、理解性の高い履歴を生成することに取り組んだ。

(2) 様々な観点での操作の特徴分析

操作履歴は、どのような観点から分析を行うかによって様々な特徴を持つ。例えば、開発者のタスクの種類(機能追加、デバッグ、リファクタリング等)によって、開発者が必要とする情報は異なるため、異なるタスクにおける操作には異なる特徴が見られることになる。本研究では、記録された操作の特徴の推移を分析することで、タスクの種類ごとの操作列の特徴を分析した。

(3) 操作履歴と抽象的情報の関連付け

OperationRecorder により得られる操作履歴は、人間が直接見ることを想定していないため、操作履歴自体からプログラム変更の内容を推測することは困難である。そこで、操作履歴の全体像やその意味を、より素早く、容易に把握できるように、操作と抽象的な情報を関連付ける手法を考案した。ここでの抽象的な情報とは、ソースコードを静的解析す

ることにより得られるプログラム構造等の情報、タスク情報である。これにより、ユーザは、記録された操作を特定のプログラム構成要素や自らの開発作業と関連付けて理解することが可能となる。

(4) 操作履歴に基づくプログラム変更支援

過去に行われた操作履歴の特徴分析に基づいて、開発者がソースコードを変更する際に、どのようなコード(識別子)を次に挿入するべきかを予測する。

４．研究成果

(1)理解しやすいプログラム変更情報の生成

OperationRecorder が出力する操作履歴情報は、開発者が実際に行った操作やソースコード変化の詳細を追跡するために有効である。しかしながら、履歴の内容について開発者が詳しくない場合等には、初めから詳細な情報を提供するよりも、履歴全体の概要をつかむために、より粗い粒度での情報を提供することが望ましいと考えられる。本研究では、履歴変換手法によって不要な操作をフィルタリングしたり、融合したりする手法について研究を行った[雑誌論文①、学会発表②]。実際の操作履歴に手法を適用した結果、大幅に操作数を減らし、効率的に操作履歴の内容を調べられるようになることが分かった。

従来、ソフトウェア進化の追跡のために、版管理システムのリポジトリに格納されたソースコードの差分を分析することが一般的であった。しかしながら、このような方法の場合、1 つのトランザクション(隣接する版の間に行われた一連の変更)において同じ箇所に対する変更が繰り返されると、個々の変更の内容が上書きされたり、混ざり合ってしまうという問題がある。そこで、操作履歴を使用することで、個々の変更を得る手法について研究を進めた[学会発表①]。この手法では、操作履歴を基に、構文要素の追加・変更・削除等のプログラム変更の情報を生成する。さらに、連続するプログラム変更を、変更の種類や、変更の時間的・空間的距離に基づくルールに従ってグループ化する。これにより、複数の変更が混ざり合うことなく、適切な粒度で理解可能になると考えられる。我々の手法では、操作のグループ化に関するパラメータは柔軟に設定することを想定している。いくつかのケーススタディによる評価実験の結果、どのようなパラメータによりグループ化するのが適切かについては、個々のプログラム理解の目的に依存することがわかった。

OperationReplayer による操作再生は、時系列ベース(つまり、操作が行われた順番に操作を適用していくこと)が基本となる。一方で、特定のクラスやメソッドに関して行われた変更を理解する場合、それ以外の箇所で行われた編集操作の再生は飛ばせるようにすることが望ましいと考えられる。そこで、編集操作が行われた箇所のオフセット値から、プログラム要素との対応をグラフ化する手法を実現した。このグラフを利用することで、特定のプログラム要素に関係する操作のみを再生することが可能となる[学会発表⑥]。

過去に行われたプログラム変更を理解する場合、個々のタスクの種類ごとに操作列が整理されていることが望ましい。しかしながら、実際には異なる目的の操作が互いに混ざり合った状態で操作列が生成される。これを解消するために、編集操作を整理する手法について研究を行った[学会発表④]。我々はこの手法を、ソースコードのリファクタリングになぞらえ、編集操作のリファクタリングと呼んでいる。具体的なリファクタリングとして、Swap、Merge、Split、Cancel、および、それらを組み合わせた Reorder や Reconfigure を提案した。さらに、編集操作のリファクタリングを支援するツールとして、Historef を実装した。Historef を使用することで、実際に行われた編集操作と、そのグループを自動的に関連付けたり、実際にリファクタリングを容易に適用することが可能となる。

以上のように、我々は複数の履歴表現手法を提案してきた。履歴の応用目的により、適切な表現手法は異なるため、今後、個々の目的に応じてこれらの履歴を適切に使い分けることで、効果的なソフトウェア開発支援を行うことができると考えられる。

(2)タスク情報と操作履歴の関連付け

OperationRecorder により記録された操作履歴情報を元に、人間にとってわかりやすい開発作業の単位として、タスクを自動的に識別する手法について研究を行った。まず、操作履歴中に含まれているメニュー機能の呼び出し 99 種類を 8 項目に分類した。次に、操作履歴を手作業で分析し、開発セッション(長期離席から次の長期離席までの期間)ごとに、各種類の操作の数とその遷移を分析した。これを元に、4 種類のタスク(EDIT, READ, DEBUG, AWAY)を定義し、自動的に操作をこれらのタスクと関連付けるツールを構築した。さらに、図 1 に示すように、OperationReplayer を拡張し、タイムライン上にタスクを色分けして表示する機能、およ

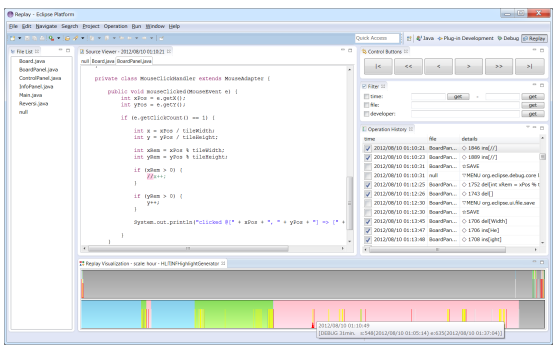

図 1 OperationReplayer におけるタスク情報表示

び、各タスクにおける編集箇所を編集頻度順に表示する機能を加えた。これにより、操作履歴再生において、各操作が行われた文脈、同時に変更された箇所を容易に把握できるようになった。提案手法がプログラム理解に関して効果的であることを示す評価実験を行う予定であったが、当初想定していたカタログ(プログラム理解に関する質問集)では、既存手法に対する優位性を示せなかったため、実験は未完了である。現在、提案手法をタスク復帰支援に適用することを想定した評価実験の準備を進めている。

(3)コード補完ツールの改善

操作履歴がソースコード変更を支援する事例として、コード補完ツールに着目した。コード補完とは、識別子の一部(または全部)やコードの断片を自動的にソースコード中に入力する IDE の機能である。まず、我々が収集した操作履歴データを分析することで、同じコード補完は短い時間のうちに繰り返されることが多いという実験結果を得た[学会発表③⑤]。コード補完の総数、再出現の回数、再出現率をグラフ化したものを図 2 に示す。横軸は比較対象となるコード補完操作からの経過時間を示しており、短い間隔(グラフ左側)ほど再出現率が高いことがわかる。

当初、この実験結果を基に、繰り返しコード補完を分析し、頻出する再出現パターンを明らかにすることで、コード変更を支援する手法を実現する予定であったが、ソフトウェア開発中にリアルタイムにパターンへの適合を判定し、挿入文字列を推測する手法を、高精度と高再現率を両立して実現することができなかった。一方、単純に最近行われたコード補完を予測順位の上位とする手法は高い性能を示した。この成果については、論文として国内論文誌に投稿した。

５．主な発表論文等

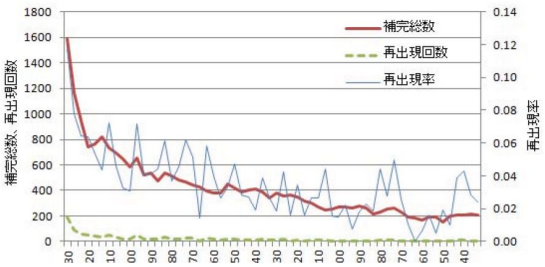

図 2 繰り返しコード補完の分布

(補完総数、再出現回数、再出現率)

（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　1 件）

①著者名：桑原寛明、大森隆行、論文表題：編集操作履歴の再生における粗粒度な再生単位、雑誌名：コンピュータソフトウェア、査読：有、巻：30、発行年：2013、ページ：61-66、https://www.jstage.jst.go.jp/article/jssst/30/4/30_4_61/_pdf

〔学会発表〕（計　6 件）

①発表者名：Eijiro Kitsu, Takayuki Omori, Katsuhisa Maruyama、発表標題：Detecting Program Changes from Edit History of Source Code、学会名等：The 20th Asia-Pacific Software Engineering Conference、発表年月日：2013 年 12 月 3 日、発表場所：Bangkok, Thailand

②発表者名：桑原寛明、大森隆行、発表標題：編集操作履歴の再生における粗粒度な再生単位、学会名等：日本ソフトウェア科学会 第 19 回 ソフトウェア工学の基礎ワークショップ、発表年月日：2012 年 12 月 15 日、発表場所：ゆふいん山水館、大分県

③発表者名：大森隆行、桑原寛明、丸山勝久、発表標題：統合開発環境におけるコード補完の繰り返しに関する調査、学会名等：日本ソフトウェア科学会 第 19 回 ソフトウェア工学の基礎ワークショップ、発表年月日：2012 年 12 月 14 日、発表場所：ゆふいん山水館、大分県

④発表者名：Shinpei Hayashi, Takayuki Omori, Teruyoshi Zenmyo, Katsuhisa Maruyama, Motoshi Saeki、発表標題：Refactoring Edit History of Source Code、学会名等：The 28th IEEE International Conference on Software Maintenance、発表年月日：2012 年 9 月 27 日、発表場所：Riva del Garda, Italy

⑤発表者名：Takayuki Omori, Hiroaki Kuwabara, Katsuhisa Maruyama、発表標題：A Study on Repetitiveness of Code Completion Operations、学会名等：The 28th IEEE International Conference on Software Maintenance、発表年月日：2012 年 9 月 25 日、発表場所：Riva del Garda, Italy

⑥発表者名：Katsuhisa Maruyama, Eijiro Kitsu, Takayuki Omori, Shinpei Hayashi、発表標題：Slicing and Replaying Code Change History to Assist Program Comprehension、学会名等：The 27th IEEE/ACM International Conference on Automated Software Engineering、発表年月日：2012 年 9 月 5 日、発表場所：Essen, Germany

〔その他〕

(1)ホームページ
http://www.fse.cs.ritsumei.ac.jp/~takayuki/

(2)立命館大学研究者学術情報データベース
http://research-db.ritsumei.ac.jp/Profiles/77/0007601/profile.html

６．研究組織

(1)研究代表者

大森　隆行（OMORI TAKAYUKI）
立命館大学・情報理工学部・助教
研究者番号：90532903