様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　３０　年　　６　月　２１　日現在

機関番号：　８２６２６
研究種目：　基盤研究(B)（一般）
研究期間：　2014～2017
課題番号：　２６２８００２５

研究課題名（和文）検索をベースとした大規模ソフトウェアの変更解析に関する研究

研究課題名（英文）A Search Based Change Analysis Method for Large-Scale Software

研究代表者
森　彰（Mori, Akira）

国立研究開発法人産業技術総合研究所・情報・人間工学領域・研究グループ長

研究者番号：３０３１１６８２

交付決定額（研究期間全体）：（直接経費）　　11,000,000円

研究成果の概要（和文）：ソースコードの構文解析木の差分を詳細に計算することで、大規模ソフトウェアのバージョンをまたいだ変更履歴を効率よく検索したり表示できるようなツールを開発した。ツールは、変更情報を格納し検索可能とするデータベースシステムや、検索された変更をソースコード上にわかりやすく表示するユーザーインターフェースを含む。このツールを用いて、実際のオープンソースの大規模プロジェクトを対象にした変更パターンの検索実験や、テスト結果が反転するソースコードの変更を自動的に同定するデバッグ手法や、さらには機械学習を用いたソースコードの意味情報抽出などの実験に取り組み、手法の有用性を示した。

研究成果の概要（英文）：We have developed a fine-grained change analysis tool for large-scale software projects based on our tree differencing algorithm. By this tool, we can efficiently search change histories across versions for a specific change pattern. The tool includes a database system that stores entire change information and the user interface that displays discovered change instances over the original source code. We have conducted a series of experiments that include fine-grained search on large-scale projects, automated debugging by way of test reversal identification, and extraction of semantic information about the source code by way of machine learning, to demonstrate the effectiveness of the method.

研究分野：　ソフトウェア工学

キーワード：　ソースコード変更解析　抽象構文木　木差分計算　RDFデータベース　SPARQL

１．研究開始当初の背景

(１) ソフトウェアの規模が大きくなり、またその開発当初から時間が経過するにつれ、その開発と維持に関わるコストが膨大なものになり、こうした作業を機械的に支援する必要があった。

(２) 開発過程のソースコードの変更を俯瞰的に理解できるような支援ツールに乏しく、従来からの行ごとの文字列の差分を表示するだけでは限界が生じていた。

２．研究の目的

(１) ソースコードの構文解析木の差分を計算するツールを活用し、大規模ソフトウェアのバージョンをまたいだ変更履歴を効率よく検索したり表示したりできるようなツールを開発すること。

(２) データベースやユーザーインターフェースを含む実用ツールを開発し、詳細な変更パターンの同定や不具合箇所の同定と修正パッチの自動生成、さらには不具合の予測などを通じて、手法の有用性を実証すること。

３．研究の方法

(１) 従来より開発を進めてきた木差分計算ツールを発展させるとともに、隣接バージョンごとの変更情報をデータベースに格納し、一括して検索、表示できるようする。

(２) 実際のオープンソースの大規模プロジェクトを対象として、詳細な変更パターンの検索実験を行い、検索文の指定方法や検索結果の集計・表示方法について検討を加え、実用可能なツール群として整備する。

(３) 高度な応用として、テスト結果が反転するソースコード編集を自動的に同定し、編集系列を修正パッチとして出力するツールの開発や、複数のプログラミング言語への対応とその類似コード検知への応用や、機械学習を用いたソースコードの意味情報抽出などに取り組む。

４．研究成果

(１)大規模ソースコードを対象として実験として、Linux カーネル 2.6.18 から 2.6.39 までの 22 個のリリースバージョンのソースコードと、Java 言語向けのビルドツールである Apache Ant の 11 から 194 までの 184 個のリリースバージョンのソースコードについて、隣接バージョン間の差分計算を行い、解析結果を RDF データベース化することを試みた。Linux カーネルについては、Coccinelle と呼ばれるパーザーが出力する命令文レベルでの制御フローグラフの情報や、Ncc と呼ばれるコンパイラーツールが出力するコールグラフの情報も付加した。Ncc ツールは、典型的な関数ポインター経由の関数呼び出しを解析することが可能であるが、開発が停止しており現在の実行環境では稼働しなかったため、不具合の修正やいくつかの改善を行った。Linux カーネルの解析結果は、合計 40 億件を超える RDF の三つ組データが生成され、これはソフトウェアの変更情報としては非常に大規模なものである。

そして、この大規模な変更履歴 RDF データベースに対して、局所変数の置き換えなどの細粒度のリファクタリングパターンを検索する実験を行った。RDF データベースに対する問い合わせ言語である SPARQL を用いて検索グラフパターンを記述して実行した結果、Linux カーネルのように大規模なデータであっても、数秒程度で数百件規模の検索結果が得られ、これは想定を上回る処理効率であった。Linux カーネルにおける Big Kernel Lock と呼ばれるロック機構に関する長期間の変更パターンを対象に、検知実験を行ったところ、正確に対象となるロック関数のペアを検知することができ、検索の規模と詳細さと精度の面で、既存手法では達成しえない性能を達成できた。全体として、スケーラビリティと精度の両面で良い成果を達成することができ、これらの研究成果は、国際会議に投稿し採択され、口頭発表を行った(学会発表③)。Java と C のように言語が異なる場合でも、ほとんど同一の検索パターンを用いることができ、ある意味でプログラミング言語の壁を超える解析手法を提示することができたことも、既存手法ではなし得ない成果であった。図１に、本研究の手法のイメージを示す。

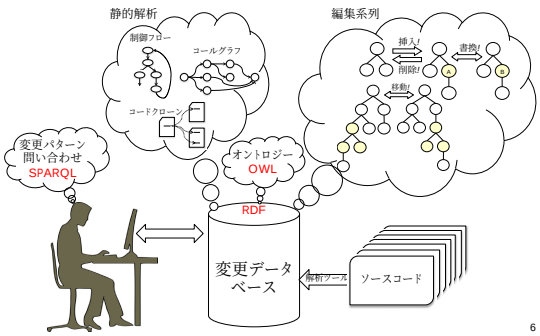

図１ 検索によるソフトウェア変更解析

(２)ソフトウェア開発にとって問題となったり、重要な設計判断を反映したりする特徴的なコード変更パターンであるリファクタリングを、効率よくかつ正確に同定するための技術について研究を行った。Java 言語によるソフトウェアプロジェクトについては、Fowler らによって整理されたオブジェクト指向プログラムにおけるリファクタリングパターンを中心に、また、C 言語によるプロジェクトについては、局所変数の導入や削除などの変更パターンを中心に、同定実験の作業を行った。全体として 60 個程度の変更パターンについて、SPARQL による問い合わせ

文を記述し、変更データベースにおいて検索を行い、検索結果が正しいかを目視で確認した。Java については、Apache Ant プロジェクトを始めとするオープンソースプロジェクトを対象に、また、C については Linux カーネル 2.6 を対象として実験を行った。結果として検出精度は 90%を超え、これは既存研究(概ね 70%程度)と比較しても非常に良好な結果である。さらに、検出された変更パターンがバージョンを超えて関連しているかを調べる実験も行った。例えば、プログラムの可読性を向上するために導入された局所変数について、その初期値が後になってインライン展開されるような逆戻り変更パターンを検出することが可能になった。これは、年月を経るにつれて、当初の変更の意図があいまいになり、一見無駄とも思える変更が生じてしまうことを意味し、ソフトウェア開発における興味深い現象を初めて明らかにする技術として非常に興味深い研究結果と言える。

(3)木差分計算により得られるバージョンごとの構文解析木の間の木編集系列を、違いに独立なグループに分割した上で二分探索を行うことにより、回帰バグの原因となったソースコード編集箇所を、ピンポイントで同定する差分デバッギングの開発に取り組んだ。分割された変更グループを選択的に適用して、仮想的な中間バージョンを構成して実行し、問題となっているテストの結果を確認していくことが必要になるが、オブジェクト指向プログラミング言語である Java を対象としたため、変更の依存関係を考慮したグループ化を行わなければ、生成された中間バージョンがビルド・実行可能なものにならないという状況が頻発した。この問題を回避するために、オブジェクト指向言語に特有のさまざまな依存性を考慮した分割ルールを定義した。数百個にも登るこうしたルールを適用することで、高い頻度でビルド・実行可能な中間バージョンを生成できるようになり、結果として、ソースコード行数 10 万行を超えるような実用規模のプロジェクトに対して、回帰バグの発生箇所を正確に同定できることがわかった。

Defects4J と呼ばれる Java プロジェクトのバグ修正パッチのデータベースを対象に、人手によるパッチ抽出と差分デバッギングによる自動抽出の結果を比べたところ、多くの場合において、人手による修正と遜色ない結果が得られた。人手による作業が信頼できない例や、バグ判定のためのテストが適切でない例も見られ、既存の不具合対策の問題点をあぶり出す結果となった。実際のプロジェクトで発生した回帰バグを修正するパッチ自動生成の実験も行い、成果をまとめてソフトウェア工学のトップ国際会議会議に投稿し採択された(学会発表①)。組織内で開発中のロボットシステムへの応用したところ、通常のテキスト差分を用いた手法であっても、Java 言語と C++言語の双方で有効であることが確認できた。また、過去の変更履歴データから、不具合修正のパターンを機械学習して、同定された不具合の修正を自動的に行う自動バグ修正技術の研究を試みた。ソースコードをテキストととらえ、自然言語処理で用いられている同意文判定の手法を適用したところ、訓練データ数が不足しているという問題はあったものの、構文的に異なっても意味的に類似しているコードを選ぶ技術として有効であることが確認できた。上述の差分デバッギングの技術を用いて、正しく機能している既存の類似コードをもとに、手元の不具合を修正することができるという点で、重要な技術を開発することができた。関連する類似コード検索の研究として、ハードウェア記述言語を対象とした実験を行い、類似コード検索の手法が、特定のプログラム言語に依存しないことを示した。この研究結果は、国内学会誌にて発表を行なった。

図 2 に差分デバッギングの手法のイメージを示す。

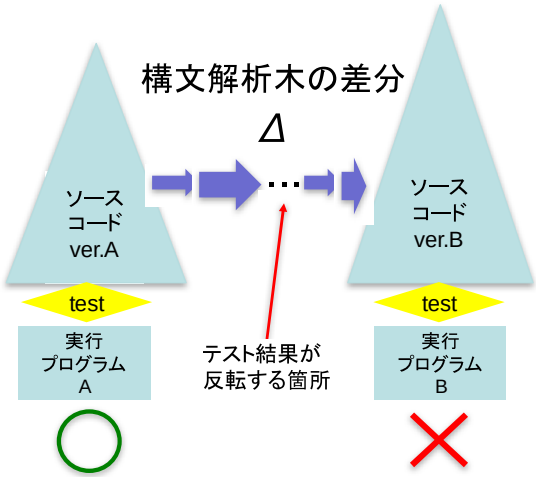

図 2　差分デバッギング

(4)開発技術の他言語への応用を試みる過程で、ハードウェア記述言語 HDL の構文解析器を開発し、その類似コード検知への応用を試みた(雑誌論文①、学会発表②)。

5．主な発表論文等

〔雑誌論文〕(計 1 件)

①上村 恭平、森 彰、藤原 賢二、崔 恩瀞、飯田 元、ハードウェア記述言語におけるコードクローンの定量的調査、情報処理学会論文誌、査読有、59 巻、4 号、2018、1225−1239

〔学会発表〕(計 3 件)

① M. Hashimoto, A. Mori, and T. Izumida, Automated Patch Extraction via Syntax- and Semantics-Aware Delta Debugging on Source Code Changes, The 26th ACM Joint

European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), Lake Buena Vista, Florida, November 4-9, 2018, To appear

② K. Uemura, A. Mori, K. Fujiwara, E. Choi, and H. Iida, Detecting and Analyzing Code Clones in HDL, In Proc of The IEEE 11th International Workshop on Software Clones (IWSC), pp. 1-7 February 2017

③ M. Hashimoto, A. Mori, and T. Izumida, A Comprehensive and Scalable Method for Analyzing Fine-Grained Source Code Change Patterns, In Proc of The IEEE 22nd International Conference on Software Analysis, Evolution and Reengineering (SANER), pp. 351-360, March 2015

６．研究組織

(1)研究代表者
森　彰（MORI, Akira）
国立研究開発法人産業技術総合研究所・情報・人間工学領域・研究グループ長
研究者番号：30311682

(2)研究分担者
橋下　政朋（HASHIMOTO, Masatomo）
千葉工業大学・人工知能・ソフトウェア技術研究センター・上席研究員
研究者番号：60357770