様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成 ２７ 年 １０ 月　 ５ 日現在

機関番号：１４４０１
研究種目：挑戦的萌芽研究
研究期間：2012～2014
課題番号：２４６５００１１

研究課題名（和文）ソースコード自動進化への挑戦

研究課題名（英文）A Challenge toward Automated Source Code Evolution

研究代表者
肥後　芳樹（Higo, Yoshiki）

大阪大学・情報科学研究科・准教授

研究者番号：７０４５２４１４

交付決定額（研究期間全体）：（直接経費）　　3,000,000円

研究成果の概要（和文）：注目しているJavaメソッドが次の変更においてどのように進化するのかを予測するために，過去に行われた変更に基づく予測モデルを提案した．提案した予測モデルでは，ソースコード中の各要素の数（例えば，二項演算の数やリターン文の数）を要素として持つベクトル型のデータを用いている．このベクトル型のデータを用いてオープンソースソフトウェアに対して実験を行い，小さい変更については75%から85%の精度で正しく予測できていた．

研究成果の概要（英文）：In this research, we have developed a model to predict the next change on a given Java method. The model is built based on the past code changes on the target software system. The model adopts a vector data, which consists of the number of every elements in the source code. For example, binomial expressions and return statements are such elements. We conducted experiments with the model and confirmed that the model was able to predict the next changes correctly with 75--85% accuracy.

研究分野：ソフトウェア工学

キーワード：リポジトリマイニング　ソースコード解析　機械学習

１．研究開始当初の背景

ソフトウェアの開発履歴に蓄えられている情報から有益なものを抽出し，今後の開発や保守作業に活かすという研究分野が注目を集めている．MSR（Mining Software Repository）といわれる分野であり，専門の国際会議が開かれている．MSR の中でも数多く行われているのはフォールトプローン（フォールトを含む）モジュールを自動特定する研究である．

ソフトウェアの保守において，ソースコードの変更作業は高い割合を占める活動である．しかし，ソースコードの変更作業は難しい．誤った方法で変更をしてしまったり，変更すべき箇所を見逃してしまったりという人的過失が発生する場合が多い．このような過失が起こってしまった場合には，追加の変更作業が必要になるだけではなく，そのような連続した変更作業自体がソフトウェアの品質に望ましくない．そのため，ソースコードの変更作業を支援し，作業の自動化を促進する技術は有益である．

２．研究の目的

究極の目標は，ソフトウェアのソースコードを自動進化させることである．ある程度開発が進んだソフトウェアを入力として与えることにより，その後の機能追加やバグ修正等のソースコードの変更を人がいっさい関わること無く，自動で行う技術を構築する．

３．研究の方法

本研究ではこの研究の第一歩として，Java メソッドを対象として，次の変更においてどのようなプログラム要素が削除および追加されるのかを予測する手法を提案する．本報告が提案する変更予測手法には下記の制限がある．

1. 変更予測の対象は Java メソッドの内部に加わる変更のみである．Java 言語以外のプログラミング言語や，Java 言語であってもクラスの追加等メソッドの外部に加わる変更は対象外である．
2. 複数のメソッドに同時に加わる修正を予測することはできない．本手法が対象にしているのは，各メソッドが次にどのように変更されるかである．
3. 本手法は変更後のソースコードを生成することはできない．本手法は，追加および削除されるプログラム要素を予測するのみである．

本研究では，変更を予測するために Java メソッドの状態ベクトルというデータ構造を提案した．状態ベクトルとは，メソッド中に存在するプログラム要素の情報を用いて作成されるベクトルである．ここでプログラム要素とは，if 文などの文の他に，識別子名，変数宣言，メソッド宣言などが含まれる．それぞれのプログラム要素の出現数が状態ベクトルの 1 つの要素となる．

本研究における実装では，プログラム要素の特定に抽象構文木を使用し，抽象構文木の頂点の種類 1 つを 1 つのプログラム要素とする．なお本実装では抽象構文木の構築に JDT(Java Development Tools) を用いた．JDT は Java を対象として抽象構文木を構築する機能を提供しており，83 種類の頂点が定義されている．本実装ではこれらの頂点のうち，コメントに関する 3 種類の頂点を除外する．従って，本実装における状態ベクトルの次元は 80 となる．

```
public int min() {
    if (x <= y) {
        return x;
    } else {
        return y;
    }
}
```

図 1 ソースコードの例

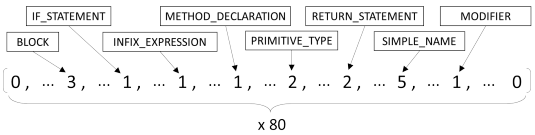

図 2 状態ベクトルの例

図 1 に簡単なメソッドのソースコードおよび図 2 にそのメソッドの状態ベクトルを示す．このメソッド中には，80 の構文要素のうち 8 のみが出現している．従って，80 の要素のうち 8 の要素は 1 以上の値を持ち，残りの 72 の要素の値は 0 となっている．なお，図 2 ではベクトルの先頭と末尾を除き，値が 0 である要素の記述を省略している．

本手法が行う処理は大きく以下の 2 つのステップから成る．

STEP1: 学習用データの抽出
STEP2: 予測モデルの構築

STEP1 は対象ソフトウェアの開発履歴情報が蓄積されたリポジトリを入力とし，予測モデルを構築するために使用する学習用データ（状態ベクトルがどのように変化したのか）を抽出する．次に STEP2 では，STEP1 で得られた学習用データをもとに，予測モデルを出力する．予測モデルに現時点でのメソッドを与えることで，その変更後の状態を予測することができる．

変更予測手法を実際にツールとして実装

した．そして複数のオープンソースソフトウェアに対して精度を調査する実験を行った．

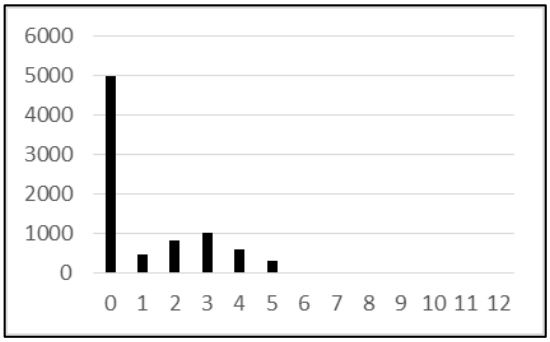

図 3 すべての要素を用いた予測結果．0 のグラフが正しく予測できた変更の数を表す．

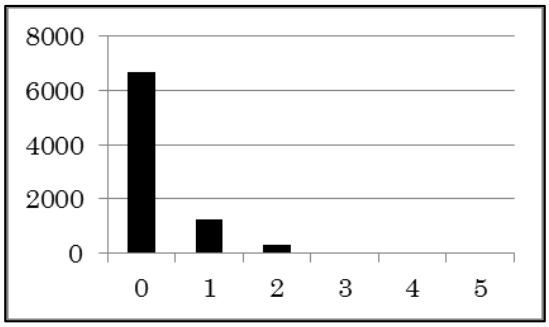

図 4 プログラム文以上の要素のみを用いた予測結果．0 のグラフが正しく予測できた変更の数を表す．

４．研究成果

本手法の予測精度は以下のとおりであった．

A. 状態ベクトルのすべての要素を用いた予測精度は 60%～70%である（図 3）．本実験では状態ベクトルのすべての要素が一致して初めて予測できたとみなしているため，60%～70%という値は高精度であると考えている．
B. 状態ベクトルのプログラム文以上の要素のみを用いた予測精度は 80%～90%である（図 4）．状態ベクトルのすべての要素を用いた予測に比べて精度が高くなった理由は，SIMPLE NAME や PRIMITIVE TYPE などの文より小さい要素が予測の対象から取り除かれたためである．SIMPLE NAME や PRIMITIVE TYPE などの要素はソースコード上において頻繁に表れるため，これらの要素の出現数を正確に予測するのは難しいと考えている．

図 3 および図 4 は，すべての要素を用いた予測結果およびプログラム文以上の要素のみを用いた予測結果の実例である．

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 3 件）

[1] 山内健二，楊嘉晨，堀田圭佑，肥後芳樹，楠本真二，“識別子名を用いたコミットのクラスタリング手法”，電子情報通信学会論文誌 D，No.66，pp.1060-1062，2015 年 6 月

[2] 今里文香，堀田圭佑，肥後芳樹，楠本真二，“機械学習を利用した危険なコードクローンの予測手法”，電子情報通信学会論文誌 D，Vol.J98-D，No.5，pp.847-850，2015 年 5 月

[3] 肥後芳樹，楠本真二，“コード修正履歴情報を用いた修正漏れの自動検出”，情報処理学会論文誌，Vol.54，No.5，pp.1686-1696，2013 年 5 月．

〔学会発表〕（計 17 件）

[1] 切貫弘之，堀田圭佑，肥後芳樹，楠本真二，“相関ルールマイニングを用いたソースコードの修正候補の推薦”，電子情報通信学会技術報告，Vol.114，No.510，pp.67-72，2015 年 3 月

[2] Hiroaki Murakami, Keisuke Hotta, Yoshiki Higo, and Shinji Kusumoto, “Predicting Next Changes at the Fine-Grained Level”, In Proc. of the 21st Asia-pacific Software Engineering Conference (APSEC2014), pp.126-133, Jedu, Korea, December 1-4, 2014.

[3] Ayaka Imazato, Keisuke Hotta, Yoshiki Higo, and Shinji Kusumoto, “Predicting Risky Clones Based on Machine Learning”, In Proc. of the 15th International Conference of Product Focused Software Development and Process Improvement (PROFES2014), pp.294-297, Helsinki, Finland, December 10-12, 2014.

[4] Kenji Yamauchi, Jiachen Yang, Keisuke Hotta, Yoshiki Higo and Shinji Kusumoto, “Clustering Commits for Understanding the Intents of Implementation”, In Proc. of the 30th International Conference on Software Maintenance and Evolution (ICSME2014), pp.406-410, British Columbia, Canada, September 28 – October 3, 2014.

[5] Yoshiki Higo, Shinji Kusumoto, “MPAnalyzer: A Tool for Finding

Unintended Inconsistencies in Program Source Code”, In Proc. of the 29th IEEE/ACM International Conference on Automated Software Engineering (ASE2014), pp.843-846, Vasteras, Sweden, September 15-19, 2014.

[6] 切貫弘之，堀田圭佑，肥後芳樹，楠本真二，“修正履歴情報を利用した Task Level Commit 支援手法の提案”，ソフトウェアエンジニアリングシンポジウム 2014，pp.85-94，2014 年 9 月.

[7] 今里文香，堀田圭佑，肥後芳樹，楠本真二，“機械学習を用いたコードクローンの危険予測手法”，電子情報通信学会技術報告，Vol.114，No.128，pp.129-134，2014 年 7 月.

[8] Hiroyuki Kirinuki, Yoshiki Higo, Keisuke Hotta, Shinji Kusumoto, “Hey! Are you Committing Tangled Changes?”, In Proc. of the 22nd International Conference of Program Comprehension (ICPC2014), pp.262-265, Hyderabad, India, June 2-3, 2014.

[9] 山内健二，楊嘉晨，堀田圭佑，肥後芳樹，楠本真二，“コード内に出現する識別子情報に基づくコミット分類”，電子情報通信学会技術研究報告，2014 年 3 月.

[10] 木村秀平，肥後芳樹，楠本真二，“リポジトリマイニングを用いたリファクタリングが開発に与える影響の測定”，電子情報通信学会技術研究報告，2014 年 3 月.

[11] Shuhei Kimura, Keisuke Hotta, Yoshiki Higo, Hiroshi Igaki, and Shinji Kusumoto, “Does Return Null Matter”, In Proc. of the IEEE CSMR/WCRE 2014 Software Evolution Week (CSMRWCRE2014), pp.244-253, Antwerp, Belgium, February 3-7, 2014.

[12] 村上寛明，堀田圭佑，肥後芳樹，楠本真二，“ソースコードの自動進化に向けて”，電子情報通信学会技術研究報告，Vol.113，No.422，pp.107-112，2014 年 1 月.

[13] Noa Kusunoki, Keisuke Hotta, Yoshiki Higo, and Shinji Kusumoto, “How Much Do Code Repositories Include Peripheral Modifications?”, In Proc. of the 5th International Workshop on Empirical Software Engineering in Practice (IWESEP2013), pp.19-24, Bangkok, Thailand, December 2, 2013.

[14] 切貫弘之，堀田圭佑，肥後芳樹，楠本真二，“ソースコード中の変数間のデータ依存関係を用いたコミットの分割”，電子情報通信学会研究報告，Vol.113，No.269，pp.67-72，2013 年 10 月.

[15] 木村秀平，堀田圭佑，肥後芳樹，井垣宏，楠本真二，“null 返り値が保守作業に与える悪影響の調査”，ソフトウェアエンジニアリングシンポジウム 2013，pp.f02:1-f02:8，2013 年 9 月.

[16] 楠野明，堀田圭佑，肥後芳樹，楠本真二，“修正の分類に基づいたコミット分割手法の提案”，電子情報通信学会技術研究報告，Vol.113，No.24，pp.31-36，2013 年 5 月.

[17] Yoshiki Higo, and Shinji Kusumoto, “How Often Do Unintended Inconsistencies Happen? -Deriving Modification Patterns and Detecting Overlooked Code Fragments-”, In Proc. of 28th International Conference on Software Maintenance (ICSM2012), pages 222-231, Italy, Riva del Garda, September 25-27, 2012.

〔図書〕（計 0 件）

〔産業財産権〕
○出願状況（計 0 件）

○取得状況（計 0 件）

〔その他〕
ホームページ等

開発した変更予測ツール
https://github.com/kusumotolab/prevol

６．研究組織
(1)研究代表者
肥後　芳樹（HIGO, Yoshiki）
大阪大学・大学院情報科学研究科・准教授
研究者番号：70452414

(2)研究分担者
なし

(3)連携研究者
なし