修士論文

ソースコードの類似性に基づいた テストコード自動推薦ツール

倉地 亮介

2020 年 1 月 28 日

奈良先端科学技術大学院大学
先端科学技術研究科 情報科学領域

本論文は奈良先端科学技術大学院大学先端科学技術研究科情報科学領域に
修士 (工学) 授与の要件として提出した修士論文である。

倉地 亮介

審査委員：

飯田 元 教授	（主指導教員）
井上 美智子 教授	（副指導教員）
市川 昊平 准教授	（副指導教員）
崔 恩瀞 准教授	（京都工芸繊維大学）

ソースコードの類似性に基づいた テストコード自動推薦ツール*

倉地 亮介

内容梗概

ソフトウェアの品質確保の要と言えるソフトウェアテストを支援することは重要である．これまでにテスト作成コストを削減するために様々な自動生成技術が提案されてきた．しかし，既存ツールによって自動生成されたテストコードはテスト対象コードの作成経緯や意図に基づいて生成されていないという性質から後のメンテナンス活動を困難にさせる課題がある．この課題の解決方法として，既存テストの再利用が有効であると考えられる．本研究ではOSSプロジェクト上に存在する既存の品質が高いテストコードを推薦するツールSuiteRecを提案する．SuiteRecは，類似コード検索ツールを用いてクローンペア間でのテスト再利用を考える．開発者からの入力コードに対して類似コードを検出し，その類似コードに対応するテストスイートを開発者に推薦する．さらに，テストコードの良くない実装を表す指標を示すテストスメルを開発者に提示し，より品質の高いテストスイートを推薦できるように推薦順位を並び替える．提案ツールの評価では，被験者によってSuiteRecを使用した場合とそうでない場合でテストコードの作成してもらい，テスト作成をどの程度支援できるかを定量的および定性的に評価した．その結果，SuiteRecを利用した場合，(1) 条件分岐が多いプログラムのテストコードを作成する際にコードカバレッジの向上に効果的であること，(2) 作成したテストコードはテストスメルの数が少なく品質が高いこと，(3) 開発者はテストの作成を容易だと認識し，自身で作成したテストコードに自信が持てることが分かった．

*奈良先端科学技術大学院大学 先端科学技術研究科 情報科学領域 修士論文, 2020年1月28日.

キーワード

類似コード検出, 推薦システム, ソフトウェアテスト, 単体テスト

Automatic Test Suite Recommendation System based on Code Clone Detection*

Ryosuke Kurachi

Abstract

Automatically generated tests tend to be less read-able and maintainable since they often do not consider thelatent objective of the target code. Reusing existing tests mighthelp address this problem. To this end, we present SuiteRec, asystem that recommends reusable test suites based on code clonedetection. Given a java method, SuiteRec searches for its codeclones from a code base collected from open-source projects,and then recommends test suites of the clones. It also providesthe ranking of the recommended test suites computed basedon the similarity between the input code and the cloned code.We evaluate SuiteRec with a human study of ten students. The results indicate that SuiteRec successfully recommends reusabletest suites.

Keywords:

clone detection, recommendation system, software testing, unit test

*Master's Thesis, Division of Information Science, Graduate School of Science and Technology, Nara Institute of Science and Technology, January 28, 2020.

目 次

1. はじめに **1**

2. 背景 **3**

2.1 ソフトウェアテスト 3

2.2 テストコード自動生成技術 4

2.3 テストスメル 5

2.3.1 テストスメルの種類 5

2.4 過去における研究 6

2.5 研究の目的と意義 6

3. 現状と今後の課題 **7**

謝辞 **8**

参考文献 **9**

付録 **10**

A. おまけその 1 **10**

B. おまけその 2 **10**

図 目 次

1　テストにおけるタスク 3
2　これは図の例 6
3　おまけの図 10

表 目 次

1　テストの種類 4
2　テストスメルの種類 11
3　これは表の例 11

1. はじめに

近年，ソフトウェアに求められる要件が高度化・多様化する一方，ユーザからはソフトウェアの品質確保やコスト削減に対する要求も増加している [1]．その中でもソフトウェア開発全体のコストに占める割合が大きく，品質確保の要ともいえるソフトウェアテストを支援する技術への関心が高まっている [?]．しかし，現状ではテスト作成作業の大部分が人手で行われており，多くのテストを作成しようとするとそれに比例してコストも増加してしまう．このような背景から，ソフトウェアの品質を確保しつつコスト削減を達成するために，様々な自動化技術が提案されている [?],[?],[?],[?],[?].

既存研究で提案されている EvoSuite[?] は，単体テスト自動生成における最先端のツールである．EvoSuite は，対象コードを静的解析しプログラムを記号値で表現する．そして，対象コードの制御パスを通るような条件を集め，条件を満たす具体値を生成する．単体テストを自動生成することで，開発者は手作業での作成時間が自動生成によって節約することができ，またコードカバレッジを向上することができる．しかし，既存ツールによって自動生成されるテストコードは対象のコードの作成経緯や意図に基づいて生成されていないという性質から可読性が低く開発者に信用されていないことや後の保守作業を困難にするという課題がある [?],[?],[?]．このことは、自動生成ツールの実用的な利用の価値に疑問を提示させる．テストが失敗するたびに，開発者はテスト対象のプログラム内での不具合を原因を特定するまたは，テスト自体を更新する必要があるかどうかを判断する必要がある．自動生成されたテストは，自動生成によって得られる時間の節約よりも読みづらく，保守作業に助けになるというよりかむしろ邪魔するという結果が報告されている [?].

我々は，この課題の解決するために既存テストの再利用が有効であると考える．本研究では，OSS に存在する既存の品質の高いテストコード推薦するツール SuiteRec を提案する．推薦手法の基本となるアイディアは類似コード間でのテストコード再利用である．SuiteRec は，入力コードに対して類似コードを検出し，その類似コードに対応するテストスイートを開発者に推薦する．さらに，テストコードの良くない実装を表す指標であるテストスメルを開発者に提示し，より品

質の高いテストスイートを推薦できるように推薦順位がランキングされる．

提案ツールの評価では，被験者によってSuiteRecの使用した場合とそうでない場合でテストコードの作成してもらい，テスト作成をどの程度支援できるかを定量的および定性的に評価した．その結果，SuiteRecの利用は条件分岐が多く複雑なプログラムのテストコードを作成する際にコードカバレッジの向上に効果的であること，作成したテストコードの内のテストスメルの数が少なく品質が高いことが分かった．また，実験後のアンケートによる定性的な評価では，SuiteRecを使用した場合被験者はテストコードの作成が容易になると認識し，また自分の作成したコードに自信が持てることが分かった．

2. 背景

2.1 ソフトウェアテスト

ソフトウェアテスト (以下，テスト) とは，ソフトウェア開発プロセスの中で最後の品質を確保する工程である．テストは，ソフトウェアが仕様書通りに動作することを確認すること，また不具合を検出し修正することでソフトウェアの品質を向上させることを目的として行われる．テストは図 [1] で示すように，テスト計画，テスト設計，テスト実行，テスト管理とうい大きく 4 つのタスクで構成される．テスト設計をさらに詳細に「テスト分析」,「テスト設計」,「テスト実装」のように分割する場合も存在するが，本研究ではテストケース作成に必要な作業をすべて「テスト設計」タスクとして扱う．テスト計画タスクでは，開発全体の計画に基づき，テスト対象，スケジュール，各タスクの実施体制・リソース配分等の策定を行う．テスト設計タスクでは，設計書などソフトウェアの仕様が記述されたドキュメント等を基に，テストケースを作成する．テスト実行タスクでは，ソフトウェアを動作させ，それぞれのテストケースにおいてソフトウェアが期待通りの振る舞いをするかどうかを確認する．テスト管理タスクでは，テストの消化状況やソフトウェアの品質状況の確認を随時行い，テスト優先度やリソース見直しなどのアクションを行う．テスト工程のコスト削減のため，テスト実行タスクにおいて，単体テストでは JUnit，結合テスト Selenium，Appium 等のテスト自動実行ツールの利用が進んでいる．しかし，テスト設計タスクは未だ手動で行うことが多く，自動化技術の実用化および普及が期待されている．

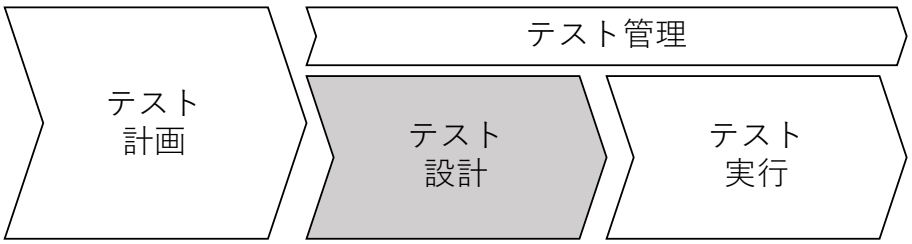

図 1 テストにおけるタスク

表 1 テストの種類

テスト粒度	説明
単体テスト	プログラムを構成する比較的小さな単位の部品が個々の機能を正しく果たしているかどうかを検証するテスト
結合テスト	個々の機能を果たすプログラムの部品 (単体) を組み合わせて，仕様通り正しく動作するかを検証するテスト
システムテスト	個々の機能や仕組みを総合した全体像のシステムとして，仕様通り正しく動作するかを検証するテスト

テスト工程は，表 1 のようにテスト対象の粒度によって単体テスト，結合テスト，システムテストの 3 種類に分類される．

単体テスト設計タスクで作成されるテストケースは，テストプロセス，テスト入力値，テスト期待値から構成される．テストプロセスに従ってテスト対象のソフトウェアにテスト入力値を与え，その出力結果をテスト期待値と比較する．これが一致していればテストは合格となり，一致しなければ不合格となる．単体テスト設計タスクにおいては多くの場合，同値分割法，境界地分析法などのテストケース作成技法を用いてテスト入力値を作成するが，ソフトウェアの要求通りに動作するかを確認するために多くのバリエーションのテスト入力値を作成する必要がある．

2.2 テストコード自動生成技術

テスト工程の支援するために様々なテストコード自動生成技術が提案されている．既存の研究 [13] は，既存のテストケースを再利用，自動生成，または再適用することによって，ソフトウェア開発のテスト工程における時間とコストを大幅に節約できることを示している．テスト生成技術は，主にランダムテスト (RT)，記号実行 (SE)，サーチベーステスト (SBST)，モデルベース (MBT)，組み合わせテストの 5 つに分類できる．SE はさらに静的記号実行 (SSE) と動的記号実行 (DSE) に分けられる．

RT とは，ソフトウェアにランダムな入力を与えるテスト手法である．無造作・均一にテストを実行するランダムテストは自動化に適しているが，コードカバレッジ率向上，バグ検出の観点において，テストケース 1 件当たりの効率は著しく悪い．

SE は対象コードを静的解析してプログラムを記号値で表現し，コード上のそれぞれのパスに対応する条件を抽出し，パスごとにパスを通るような入力値が満たすべき条件を集める．そして，パスごとにその条件を SMT ソルバ [5] などの制約ソルバを用いて解き，得られた具体値をテスト入力値とする．

SBST は，達成したい要件に対する達成度合いを定量的に評価できるように設計した評価関数に基づいて，ヒューリスティック探索アルゴリズムを用いて達成したい要件を満足するテストスイートを生成する技術の総称である．

MBT はモデルに基づいてテストスイートを生成する技術の総称である．モデルは何らかの形でテスト対象を記述したものであり，要求分析や設計のためのモデルを活用することもあれば，テストのためにモデルを作成することもある．

CT は，パラメータ間の相互作用に起因する不具合を効果的に発見するためにテストケースとしてパラメータに割り当てる値の組み合わせを生成する手法である．

2.3 テストスメル

テストスメルとは，テストコードの良くない実装を示す指標である．プロダクションコードの設計だけでなく，テストコードを適切に設計することの重要性は元々Beck ら [] によって提唱された．さらに，Van Deursen ら [50] は 11 種類のテストスメルのカタログ，すなわちテストコードの良くない設計を表す実装とそれらを除去するためのリファクタリング手法を定義した．このカタログはそれ以降，18 個の新しいテストスメルを定義した Meszaros [42] によってより拡張された．

2.3.1 テストスメルの種類

この節では，本研究で扱う 6 種類のテストスメルを紹介する．

ここに図を書く

図 2　これは図の例

2.4 過去における研究

2.5 研究の目的と意義

3. 現状と今後の課題

謝辞

Thank you. Thank you.

参考文献

[1] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25(NIPS'12)*, pages 1097–1105, 2012.

これはおまけの図です。

図 3 おまけの図

付録

A. おまけその１

B. おまけその２

表 2 テストスメルの種類

テストスメル	説明
Assertion Roulette	テストメソッド内に複数の assert 文が存在する．各 assert 文は異なる条件をテストするが，開発者へ各 assert 文のエラーメッセージは提供されない，そのため assert 文の 1 つが失敗した場合，失敗の原因を特定するが困難である．
Default Test	テストメソッド名が初期状態 (意味のない名前) である．テスティングフレームを使用した場合，クラス・メソッド名が初期状態である．テストコードの可読性向上のために適切な名前に変更する必要がある．
Conditional Test Logic	テストメソッド内に複数の制御文が含まれている．テストの成功・失敗は制御フロー内にある assert 文に基づくためテスト結果を予測できない．また，条件分岐が多く複雑なテストコードは可読性を下げる．
Eager Test	テスト対象クラスの複数のメソッドを呼び出す．1 つのテストメソッドで複数のメソッドを呼び出すと，他の開発者は何をテストしているかについて混乱が生じる．
Exception Handling	テストメソッド内に例外処理が含まれている．例外処理は，対象コードに記述すべきで，テストコード内では正しく例外処理が行われるかを確認すべきである．
Mystery Guest	テストメソッド内で，外部リソースを利用する．テストメソッド内だけでなく外部ファイルなど，外部リソースを使用すると見えない依存関係が生じる．何らかの影響で外部ファイルを削除されるとテストが失敗してしまう．

ここに表を書く

表 3 これは表の例