社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

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 SuiteRec

倉地　亮介† 　崔　　恩瀞†† 　飯田　　元†

† 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　〒630–0192 奈良県生駒市高山町 8916–5
†† 京都工芸繊維大学情報工学課程　〒606–8585 京都府京都市左京区松ケ崎橋上町
E-mail: †kurachi.ryosuke.kp0@is.naist.jp, ††echoi@kit.ac.jp, †††iida@itc.naist.jp

あらまし　テスト工程において，テスト作成コストを削減するために様々なテストコード自動生成ツールが提案されてきた．しかし，既存のツールによって生成される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いないという性質から開発者の保守作業を困難にする課題がある．この課題の解決方法として，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．また，被験者実験を行い SuiteRec の有用性を確認した．

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，テストスメル，単体テスト

SuiteRec: Automatic Test Suite Recommendation System based on Code Clone Detection

Ryosuke KURACHI†, Eunjong CHOI††, and Hajimu IIDA†

† Graduate School of Information Science, Nara Institute of Science and Technology　8916–5, Takayama, Ikoma, Nara, 630–0192,Japan
†† Department of Information Science　Matsugasaki, Sakyo-ku, Kyoto, 606–8585 Japan
E-mail: †kurachi.ryosuke.kp0@is.naist.jp, ††echoi@kit.ac.jp, †††iida@itc.naist.jp

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 might help 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 code clones 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 based on 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 reusable test suites.

Key words　clone detection, recommendation system, software testing, test smell, unit test

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 [?]．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テスト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大幅に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後の保守作業を困難にするという課題がある [?], [?], [?]．これは，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コード内

で不具合の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コードは，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，開発者の助けになるというよりかむしろ困難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クローンペア間でのテストコード再利用である．SuiteRec は，入力コード片に対して類似コード片を検出し，その類似コード片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．

評価実験では，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に含まれる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，SuiteRec は開発者が参考にしたいテストコードを上位に推薦できることを確認した．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．

以降，2 章では，本研究に関わるコードクローン及びソフトウェアテストについての背景を説明する．3 章では，本研究で提案するテストコード自動推薦ツール SuiteRec ついて説明する．4 章では，被験者による SuiteRec の評価実験について説明する．5 章では，SuiteRec の有効性と妥当性への脅威について議論する．最後に，6 章でまとめと今後の課題について説明する．

2. 背　　景

本節では，テストコードの品質を定量的に測定する基準の 1 つであるテストスメルと既存のテストコード自動生成技術ついて説明する．

2.1 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，Deursen ら [?] 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，19 個の新しいテストスメルを定義した Meszaros [?] によって拡張された．

以下に，本研究で扱う 6 種類のテストスメルを説明する．

Assetion Roulette: テストメソッド内に複数の assert 文が存在するテストコード. 各 assert 文は異なる条件をテストするが，テストが失敗した場合開発者へ各 assert 文のエラーメッセージは提供されないので，失敗を特定することが困難になる．

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スイート．テスト成功・失敗は制御フロー内にある assert 文に基づくの予測するのが難しい．

Default Test: JUnit などのテスティングフレームワークを使用したテストコードの内，テストクラスやテストメソッドの名前がデフォルトの状態であるテストコード．テストコードの可読性の向上ために適切な名前に変更する必要がある．

Eager Test: テスト対象クラス内の複数のメソッドを呼び出しているテストコード．テストメソッド内で複数のメソッド呼び出しを行うと，何をテストしているかについて混乱が生じる．

Exception Handling: テストメソッド内で例外処理が含まれているまたは，例外を投げるテストコード．例外処理は対象コードに記述し，テストコード内で例外処理が正しく行われるかどうかを確かめるようにリファクタリングする必要がある．

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コード．テストメソッド内だけで完結せず外部のファイルなど，外部リソースを利用すると外部との依存関係が生じ，外部リソースが壊れた場合テストも失敗してしまう．

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．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．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，SBST を実装したツールであり，2011 年に発表されて以来 EvoSuite をベースとした数多くの研究がなされている．単体テストを自動生成することで，開発者は手作業でのテスト作成時間を節約することができ，またコードカバレッジを大幅に向上することができる．しかし，EvoSuite などの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ので，開発者の保守作業を困難にするという課題がある [?], [?], [?]．開発者は，テストが失敗するたびに，テスト対象のコード内で不具合の原因を特定するまたは，テスト自体を更新するか否かを判断する必要がある．Shamshiri らは，自動生成されたテストコードは可読性が低く，開発者がテスト対象コードの不具合を特定するのに，効果的でないことを報告した． また，Palomba ら [20] は，EvoSuite によって自動生成されたテストコードは，プロジェクト内にテストスメルを拡散させることを確認した．自動生成ツールによって大量に生成されるテストコードは，プロジェクト内にテストスメルを拡散させ，開発者の可読性と保守性に大きな影響を与える可能性がある．

3. SuiteRec: テストコード自動推薦ツール

本章では，本研究で提案するコードクローン検出技術を用いたテストコード自動推薦ツール SuiteRec について述べる．本研究では，コードクローン検出技術を利用し，オープンソースソフトウェア (以下，OSS) 上に存在する既存の高品質なテスト

コード推薦することで，開発者のテストコード作成を支援することが目的である．SuiteRec の基となるアイディアは，クローンペア間でのテストコード再利用である．SuiteRec は，開発者からの関数単位の入力コード片に対してその類似コード片を検出する．そして，類似コード片に対応するテストスイートを開発者に推薦する．さらに，推薦されるテストスイート内に含まれるテストスメルを提示し，より品質が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．

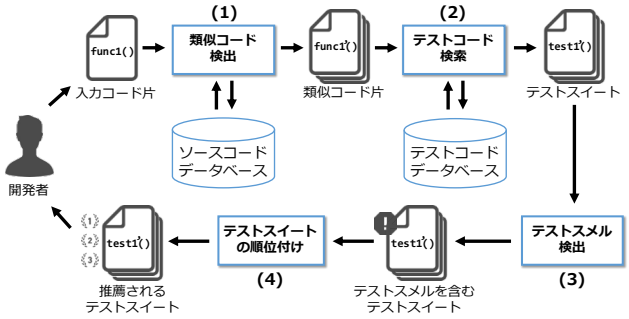

図 1　Overview of SuiteRec.

(1)　SuiteRec は，入力されたコード片を受け取ると，そのコード片をコードクローン検索ツールにかけ入力コード片の類似コードを検出する．

(2)　複数の類似コード片が検出されると，次にその類似コード片に対応するテストスイートをテストコードリポジトリ内から検索する．

(3)　各類似コード片のテストスイートが検出されると，次にそれらをテストスメル検出ツールにかけ各テストスイートに含まれるテストスメルを検出する．

(4)　最後に，(1) で得られた類似コードと入力コードの類似度と (3) で検出されたテストスメルの数を基に出力されるテストスイートの順番がランキングされる．

3.1　Step1: 類似コード片の検出

Step1 では，開発者から与えられた関数単位のコード片に対して類似コード片を検出する．本研究では，コードクローン検出ツールとして NiCad [?] を採用した．NiCad は，検出対象のコード片のレイアウトを統一的に変換させ，行単位で関数単位のコード片を比較することで，高精度・高再現率でクローンペアの検出するツールである．

本研究で，NiCad を採用した理由として以下の 2 点が挙げられる．

- 単体テストの再利用を考える上で関数単位のコード片を高精度で検出できる
- 構文的に類似した type2，type3 のコードクローンを高速に検出できる

NiCad は，入力として与えられたプロジェクトに対して，そのプロジェクト内のすべてのコードクローンを検出する．検出されるコードクローンはクローンクラスとして出力される．SuiteRec は，NiCad の検出結果の中で入力コード片を含むクローンクラスを特定し，そのクローンクラスに含まれるコード片をすべて類似コード片として扱う．

SuiteRec は，NiCad を用いて入力コード片に対する類似コード片を大規模な OSS プロジェクトを保持するソースコードデータベースから検索する．ソースコードデータベースには，テストコードが存在する Github(注1) 上の 3,205 個の OSS プロジェクトのプロダクションコードが格納されている．具体的には，既存のコード検索エンジンで利用されたデータセット [?] の内，テストフォルダが存在し，JUnit のテスティングフレームワークを採用しているプロジェクトを選択した．

NiCad は，一度に検索できるプロジェクトの規模限度がある．そこで，本研究では検索時間を短縮するために，検索対象のプロジェクトに前処理を行い，ソースコードデータベースに格納した．具体的には，大規模なプロジェクトは分割し，小規模なプロジェクトは統合させた状態で検索処理を実行した．さらに，検索処理を複数のプロジェクトに対して並列して走らせることで，現実的な時間での類似コード片の検索を実現した．また，NiCad に与えるパラメータには，検出粒度，類似度の閾値，そして検出されるコードクローン行数の閾値があり，本研究では，NiCad の標準設定のパラメータである，検出粒度: 関数，類似度の閾値: 0.3，最小行数: 5 行という検出設定で SuiteRec に実装した．

3.2　Step2: テストコードの検索

Step2 では，Step1 で検出された類似コード片に対するテストスイートを検索する．大規模なプロジェクトのテストコードが格納されているテストコードデータベース (以下，TDB) からテストスイートを検出するために，テスト対象コードとテストコードの対応付けを行う．

本研究では，対象コードとテストコードを対応付けるために以下の 3 つのフェーズを実施する．

Phase1
命名規則によるクラス単位で，テストクラスとテスト対象クラスを対応付ける．

Phase2
テストコードを静的解析し，各テストケースから呼び出されるすべてのプロダクションコードのメソッド名を収集する．

Phase3
テストメソッド名を区切り文字や大文字で分割し，テスト対象のメソッド名と部分一致した場合，テストコードとテスト対象コードをメソッド単位で対応付ける．

本研究は，JUnit テスティングフレームワークを用いた単体テストを対象とする．Phase1 では，JUnit の命名規則に従ってテストクラス名の先頭または，末尾に “Test” という文字列が含まれるテストクラスを収集し，収集したテストクラスから “Test” を除いたクラス名をテスト対象クラスとする．例えば，テスト対象クラスである “Calculator クラス” とテストクラスである “CalculatorTest クラス” が対応付けられる．

Phase2 では，テストコードを ANTLR(注2) を用いて静的解析

(注1)：https://github.com/
(注2)：https://www.antlr.org/

し，テストメソッド内で呼び出されるテスト対象のメソッド名を取得する．単体テストは，一般的に図 2 の例のようにテストコード内でテスト対象のオブジェクトの生成を行い，テスト対象のメソッド呼び出して実行される．すなわち，TCD 内のテストコードを静的解析し，テスト対象のメソッド呼び出しを取得することで，テスト対象コードとテストコードを対応付ける．ただし，テストメソッド内では，複数のテスト対象のメソッドが呼び出されていることも考えられるので，Phase3 では，さらにテスト対象のメソッド名とテストメソッド名の比較も行う．テストメソッド名の記述方法としてテスト対象メソッドの処理の内容を忠実に表すことが推奨されており，テストメソッド名にテスト対象メソッドの名前が記述されていることが多い [?]．従って，テストメソッドの名前を区切り文字や大文字で分割し，テスト対象のメソッド名と部分一致した場合，テストコードとテスト対象コードをメソッド単位で対応付けるように実装した．

```
public class Calculator {

    public int multiply(int x, int y) {
        return x * y;
    }

}
```

テスト対象コード

```
@Test
public void testMultipyOfTwoNumbers() {
    Calculator calc = new Calculator();
    int expected = 200;
    int actual = calc.multiply(10,20);
    assertEquals(expected,actual);
}
```

テストコード

図 2　テストコードと対象コードの対応付け

3.3　Step3: テストスメルの検出

Step3 では，Step2 で検索されたテストスイートに対して，テストスメルを検出する．本研究では，テストスメル検出ツールとして tsDetect(注3) [?] を採用した．tsDetect は AST ベースの検出手法で実装されたツールであり，19 種類のテストスメルを検出できるツールである．また，85%〜100% の検出精度と 90%〜100% の再現率でテストスメルを検出できる．tsDetect は，既存研究で利用実績があり高精度で多くのテストスメルを検出できるので本研究でも利用した．本研究では，tsDetect で検出できる 19 種類のテストスメルの内，**??**節で説明したテストコードの推薦を考える上で重要な 6 種類のテストスメルを提示するように実装した．

tsDetect は，AST ベースの検出手法であり大規模なテストコードに対して実行すると計算コストが高い．そのため，TDB 内のテストコードに対してテストスメルを検出し，その情報をテストコードに対応付けて TDB に格納した．これにより，推薦プロセスにおけるテストスメル検出時間を短縮化した．また，再利用対象のテストコードとして相応しくない以下の 4 つのテストスメルを含むテストコードを事前に TDB から除去し，SuiteRec が推薦するテストコードとして出力されないようにした．

- **Empty Test**：テストメソッド内にテストの記述が無く，コメントのみが含まれているテストコード

(注3)：https://testsmells.github.io/

- **Ignored Test**：@Ignore アノテーションがあり，実行されないテストコード
- **Redundant Assertion**：必ずテストが成功する意味のないテストコード
- **Unknow Test**：assert 文が存在しないテストコード

3.4　Step4: 推薦されるテストスイートの順位付け

最後の Step4 では，開発者が参考にしたいテストスイートを上位に提示できるようにテストスイートの並び替えを行う．

SuiteRec の順位付けは，Step1 の入力コード片と類似コード片の類似度と Step3 で検出されたテストスメルの数を基に計算する．我々は，以前の調査 [?] で，クローンペア間とテストコードの類似度の関係を調査した．具体的には，OSS 上に存在する 3 つの有名 Java プロジェクト内にある両方のコード片にテストコードが存在するコードクローンを対象に，クローンペア間の類似度とそれぞれのコード片に対するテストコードの類似度の関係を調査した．その結果，テストコード間の類似度と対象のクローンペア間の類似度には相関関係があり，クローンペア間の類似度が高いほどテストコードを再利用できる可能性が高いことが分かった．SuiteRec では，この結果を基に類似度が高いクローンペアの順にテストスイートを並び替え，さらに類似度が高い場合，テストスメルの数で推薦するテストスイート順位を決定するように推薦ランキングを実装した．

本研究では，クローンペア間の類似度を NiCad で用いられる計算方法である *Unique Percentage of Items* (以下，*UPI*) を用いて計算する．以下に，*UPI* の計算式を示す．

$$Unique\ Percentage\ of\ Items\ (UPI) = \frac{No.\ of\ Unique\ Items * 100}{Total\ No.\ of\ Items}$$

ここで，*No.of Unique Items* は，入力コード片と類似コード片を行単位で比較した際に一致した行の数を意味する．*Total No.of Items* は，比較対象となる各コード片内のすべての行の数を意味する．比較対象のコード片の行数が異なる場合，行数が大きい方の *Total No.of Items* を用いて計算する．例えば，入力コード片の行数が 10，類似コード片の行数が 12 で，一致した行数が 9 の場合，入力コード片と類似コード片の *UPI* は 75% になる．

4.　評 価 実 験

5.　議　　論

6.　まとめと今後の課題

文　　献

[1] D.E. クヌース，改訂新版 TEX ブック，アスキー出版局，東京，1992.
[2] 磯崎秀樹，LATEX 自由自在，サイエンス社，東京，1992.
[3] S. von Bechtolsheim, TEX in Practice, Springer-Verlag, New York, 1993.
[4] 藤田眞作，化学者・生化学者のための LATEX—パソコンによる論文作成の手引，東京化学同人，東京，1993.
[5] 阿瀬はる美，てくてく TEX，アスキー出版局，東京，1994.

[6] N. Walsh, Making TEX Work, O'Reilly & Associates, Sebastopol, 1994.
[7] D. Salomon, The Advanced TEX book, Springer-Verlag, New York, 1995.
[8] 藤田眞作，LATEX マクロの八衢，アジソン・ウェスレイ・パブリッシャーズ・ジャパン，東京，1995.
[9] 中野賢，日本語 LATEX 2ε ブック，アスキー出版局，東京，1996.
[10] 藤田眞作，LATEX 2ε 階梯，アジソン・ウェスレイ・パブリッシャーズ・ジャパン，東京，1996.
[11] 乙部巌己, 江口庄英, pLATEX 2ε for Windows Another Manual，ソフトバンク パブリッシング，東京，1996–1997.
[12] ポール W. エイブラハム，明快 TEX，アジソン・ウェスレイ・パブリッシャーズ・ジャパン，東京，1997.
[13] 江口庄英，Ghostscript Another Manual，ソフトバンク パブリッシング，東京，1997.
[14] マイケル・グーセンス，フランク・ミッテルバッハ，アレキサンダー・サマリン，LATEX コンパニオン，アスキー出版局，東京，1998.
[15] ビクター・エイコー，TEX by Topic—TEX をよく深く知るための 39 章，アスキー出版局，東京，1999.
[16] レスリー・ランポート，文書処理システム LATEX 2ε，ピアソンエデュケーション，東京，1999.
[17] 奥村晴彦，[改訂版] LATEX 2ε 美文書作成入門，技術評論社，東京，2000.
[18] マイケル・グーセンス，セバスチャン・ラッツ，フランク・ミッテルバッハ，LATEX グラフィックスコンパニオン，アスキー出版局，東京，2000.
[19] マイケル・グーセンス，セバスチャン・ラッツ，LATEX Web コンパニオン—TEX と HTML/XML の統合，アスキー出版局，東京，2001.
[20] ページ・エンタープライゼス(株)，LATEX 2ε マクロ & クラスプログラミング基礎解説，技術評論社，東京，2002.
[21] 藤田眞作，LATEX 2ε コマンドブック，ソフトバンク パブリッシング，東京，2003.
[22] 吉永徹美，LATEX 2ε マクロ & クラスプログラミング実践解説，技術評論社，東京，2003.
[23] https://oku.edu.mie-u.ac.jp/~okumura/texwiki/

付　　録

1. PDF の作成方法と A4 用紙への出力

- PDF に書き出すには二通りの方法があります．

（1） dvipdfmx を使って PDF に変換する．

```
dvipdfmx -p a4 -x 1in -y 1in -o file.pdf file.dvi
```

オプションの `-p a4 -x 1in -y 1in` は省略できます．

（2） まず，dvips を使用して，ps に書き出します（以下では段幅の関係で折り返します）．

```
dvips -Pprinter -t a4 -O 0in,0in
 -o file.ps file.dvi
```

`printer` には，使用するプリンタ名を記述します．オプションの `-t a4 -O 0in,0in` は省略できます．

次に Acrobat Distiller で PDF に変換します．

- `dvips` を使用して A4 用紙に出力する場合のパラメータはおおよそ以下のような設定になります．

```
dvips -Pprinter -t a4 -O 0in,0in file.dvi
```

`printer` には使用するプリンタ名を記述します．オプションの `-t a4 -O 0in,0in` は省略できます．

2. 削除したコマンド

本誌の体裁に必要のないコマンドは削除しています．削除したコマンドは，`\part`，`\theindex`，`\tableofcontents`，`\titlepage`，ページスタイルを変更するオプション（`headings`，`myheadings`）などです．