# 複数のコードクローン検出ツールによって検出されるコードクローンの差異を用いた不具合検出手法

澤健一 † 肥後芳樹 † 楠本真二 †

† 大阪大学 大学院情報科学研究科

**あらまし**

これまでに多くのコードクローン検出ツールが開発されているが，それらはどれも異なったコードクローンの定義を用いているため，同じ対象からコードクローンを検出した場合でも検出結果が異なる．本稿では，その違いを用いることにより，特定の不具合を検出する手法を提案する．

## 1 まえがき

近年，ソフトウェアの保守を困難にさせる要因として，コードクローンに対する関心が高まってきている．コードクローンとは，ソースコード上に存在する同一もしくは類似したコード片のことであり，コピーアンドペースト等によって発生する [1]．あるコード片にバグが存在した場合や，保守変更が必要になった場合には，そのコードクローン全てに対して同様の修正を行わなければならない．ソフトウェアの規模が拡大するほど，この修正に要する作業量が増大する．

そのため，コードクローンを検出する様々な手法が提案されている．それら既存のコードクローン検出ツールは，どれも異なるコードクローンの定義を用いているため，同じ対象からコードクローンを検出した場合でも，ツール毎に検出結果が異なる．

本稿では，それらの違いを利用して各ツールの検出結果の差分を取得することで，特定のコードクローン情報のみを取得し，不具合の検出を容易にする手法を提案する．

## 2 コードクローンの分類と特徴

### 2.1 コードクローンの分類

Bellon らは，コードクローンを次の 3 つのタイプに分類している [2]．

タイプ 1：空白，改行位置など，コーディングスタイルのみが異なるコードクローン

タイプ 2：変数名，関数名や変数の型など，識別子のみが異なるコードクローン

タイプ 3：文の挿入，削除，変更が行われたコードクローン

本稿では，このうちタイプ 1，タイプ 2 をさらに分け，5 つのタイプに分類する．

タイプ 1-1：表面上完全に同一なコードクローン

タイプ 1-2：空白，改行位置に違いがあるコードクローン

タイプ 2-1：パラメタライズドマッチング（P-Match）であるコードクローン

タイプ 2-2：P-Match でないコードクローン

タイプ 3：文の挿入，削除，変更が行われたコードクローン

表 1: 各検出手法によって検出されるタイプ

| 検出手法 | 検出されるタイプ | | | | |
|---|---|---|---|---|---|
| | 1-1 | 1-2 | 2-1 | 2-2 | 3 |
| Ducasse の手法 [3] | ○ | ○ | × | × | × |
| Baker の手法 [4] | ○ | ○ | ○ | × | × |
| Kamiya の手法 [1] | ○ | ○ | ○ | ○ | × |
| Merlo の手法 [5] | ○ | ○ | ○ | ○ | ○ |

**表 1** に既存のコードクローン検出ツールがどのタイプのコードクローンを検出できるのかをまとめる．各ツールによって検出することのできるコードクローンのタイプが異なっているのが分かる．

### 2.2 パラメタライズドマッチング（P-Match）

P-Match とは，ソースコードの比較を行うときに，識別子の対応付けを行うものである．**図 1** に P-Match の例を示す．P-Match を用いている場合，比較を行うときに識別子を区別するので，**図 1** の code1 と code2 はコードクローンでないと判定される．一方，P-Match を用いない場合は，識別子を区別しないので，コードクローンであると判定される．P-Match でないコードクローンには意味的な違いが含まれているため，P-Match であるコードクローンに比べて，不具合を含んでいる

---

**An Approach to Find Bugs Using Differences of Code Clones Detected by Different Code Clone Dection Tools**

Kenichi SAWA†, Yoshiki HIGO†and Shinji KUSUMOTO†

†Graduate School of Information Science and Technology, Osaka University

1-3, Machikaneyama-cho, Toyonaka, Osaka, 560-8531, Japan

{k-sawa, higo, kusumoto}@ist.osaka-u.ac.jp

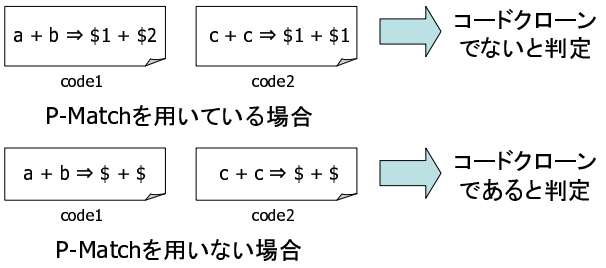


図 1: P-Match の例

可能性が高いと言えるであろう．

### 2.3 各タイプの特徴

各タイプに分類されたコードクローンはそれぞれ特徴が異なっており，起こりうる問題点にも差異が生じる．

タイプ 1-2：意味的には同一であるが，見た目に違いがあり，可読性が低い可能性が考えられる．

タイプ 2-2：変数名，関数名や変数の型などが変更されているが，P-Match になっていないため，識別子を変更し忘れている可能性が考えられる．

タイプ 3：新たな文の挿入，不要な文の削除，文の変更が行われているため，文の挿入，削除，変更し忘れの可能性が考えられる．

## 3 提案手法

### 3.1 基本方針

本稿で提案する手法は，各タイプごとの特徴に注目し，特定のタイプのコードクローンのみを出力することで，不具合の検出を容易にするというものである．

### 3.2 概要

**表 1** で示したように，既存のコードクローン検出ツールは異なったタイプのコードクローンを検出する．それら異なるコードクローン検出ツールの差分を取ることで，特定のタイプのコードクローン情報のみを得ることができる．

例えば，Merlo の手法 [5] と Kamiya の手法 [1] の出力結果の差分を取ることで，タイプ 3 のコードクローンのみを取得できる．また，Kamiya の手法と Baker の手法 [4] の出力結果の差分を取ることで，タイプ 2-2 のコードクローンのみを取得できる．

差分の取得は，あるコードクローンが，2 つのコードクローン検出ツールいずれでも検出された場合にのみ行う．検出されたコードクローンが同一であり，差分を取得すると判断するためには，コードクローンがソースコード上の位置においてどの程度一致していれば良いかを考える必要がある．Baker は，2 つのコードクローンが 70 %以上一致していれば，それらが同一であると見なすと良いだろうと述べている [4]．提案手法を実現するツールでも，この値を基準にする予定である．

## 4 まとめ

本稿では，コードクローン検出ツールの出力結果の差分を取ることで，分類した各タイプのコードクローンのみを取り出し，不具合の検出を容易にする手法を提案した．

既存のコードクローン検出ツールの出力結果のみを使用した場合は，あるコードクローンにどのような不具合の可能性が高いかが分からなかった．しかし，提案手法を用いることで，特定の不具合を含むコードクローンのみを検出することができ，結果として，そのコードクローンを調査する時間的コストの削減が見込めると考えられる．

今後の課題として，差分を取得するツールの作成を行い，再現率や適合率などを用いた評価を行う予定である．

## 謝辞

本研究は一部，文部科学省リーディングプロジェクト「e-Society 基盤ソフトウェアの総合開発」の委託に基づいて行われた．また，IJARC(マイクロソフト産学連携機構) 第 3 回 CORE プロジェクト，文部科学省科学研究費補助金 基盤研究 (A)(課題番号:17200001)，および若手研究 (スタートアップ)(課題番号:19800022) の助成を得た．

## 参考文献

[1] T. Kamiya, S. Kusumoto, and K. Inoue. Ccfinder : A multi-linguistic token-based code clone detection system for largescale source code. *IEEE Transactions on Software Engineering*, Vol. 28, No. 7, pp. 654–670, Jul 2002.

[2] S. Bellon, R. Koschke, G. Antoniol, J. Krinke, and E. Merlo. Comparison and evaluation of clone detection tools. *IEEE Transactions on Software Engineering*, Vol. 33, No. 9, Sep 2007.

[3] S. Ducasse, M. Rieger, and S. Demeyer. A language independent approach for detecting duplicated code. *Proc. of the International Conference on Software Maintenance*, pp. 109–118, Aug 1999.

[4] B.S. Baker. Finding clones with dup : Analysis of an enperiment. *IEEE Transactions on Software Engineering*, Vol. 33, No. 9, Sep 2007.

[5] J. Mayrand, C. Leblanc, and E.M. Merlo. Experiment on the automatic detection of function clones in a software system using metrics. *Proc. of the International Conference on Software Maintenance*, pp. 244–254, Nov 1996.