様式 C-19

科学研究費補助金研究成果報告書

平成２１年５月１４日現在

研究種目：若手研究（スタートアップ）
研究期間：2007～2008
課題番号：19800022
研究課題名（和文） コードクローン分析手法の実用化に関する研究

研究課題名（英文） Research on Practical Realization of Code Clone Analysis

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・助教
研究者番号：70452414

研究成果の概要：本研究では，主に，「ギャップを含むコードクローンの特定」と「バグを含んでいる可能性の高いコードクローンの特定」を行ってきた．研究期間内では，オープンソースソフトウェアのみに対して実験を行うことができたが，非常に良い結果が得られた．今後はこの研究をより発展させ，商用のソフトウェア開発・保守において利用できるよう改良を行っていきたい．

交付額

（金額単位：円）

	直接経費	間接経費	合 計
2007 年度	1,360,000	0	1,360,000
2008 年度	1,310,000	393,000	1,703,000
年度			
年度			
年度			
総 計	2,670,000	393,000	3,063,000

研究分野：総合領域
科研費の分科・細目：情報学・ソフトウェア
キーワード：ソフトウェア工学

1．研究開始当初の背景

近年，ソフトウェアの保守を困難にしている要因の１つとしてコードクローンが注目されている．コードクローンとは，ソースコード中の或る一部分（コード片）のうち，他のコード片と同一または類似しているものを指す．コードクローンはコピーアンドペーストなどによりソースコード中に作り込まれる．コピー元のコード片にバグが含まれていた場合，そのコピー先の全てのコード片（コードクローン）について修正の是非を考慮しなければならない．そのため，コードクローンを検出し，その状態を把握することが効率的なソフトウェア保守を行うために必要とされている．

これまでにさまざまなコードクローン検出手法が提案されている．報告者が所属する研究グループでも検出ツール CCFinder[1]を開発してきている．CCFinder はスケーラビリティが高く，数百万行単位の（実際に社会で利用されている規模の）ソースコードであっても数十分ほどで検出処理を完了することができる．

また，コードクローンの可視化や理解支援の研究も行われている．ツールによるコードクローン検出は，膨大な量のコードクローンを検出してしまう傾向にあるため，１つ１つを

確認することは現実的ではない．そのため，分析者が必要なコードクローンのみをわかりやすく提示する必要がある．これまでにいくつかの可視化手法が提案されているが，その中でも散布図がよく用いられている（図 1）．

散布図の水平方向と垂直方向には対象システムのソースコード中に含まれる字句が出現順に並んでいる．散布図の各要素において，その水平方向と垂直方向の成分が等しい場合に，点がプロットされる．つまり検出されたコードクローンは，ある一定以上の長さをもった線分として出現する．散布図を用いることにより，対象システム中にどのようにコードクローンが分布しているのかを知ることができる．報告者も，本研究以前にに，散布図を搭載したコードクローン可視化ツール Gemini を開発し，商用を含む様々なソフトウェアに対して適用してきた[2]．

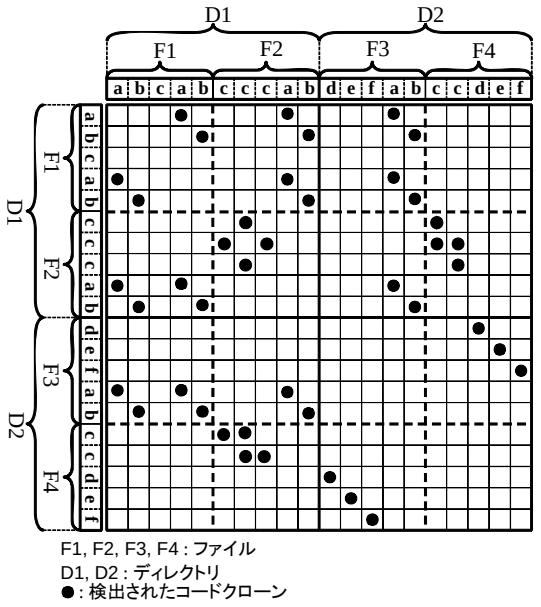

図 1 散布図モデル

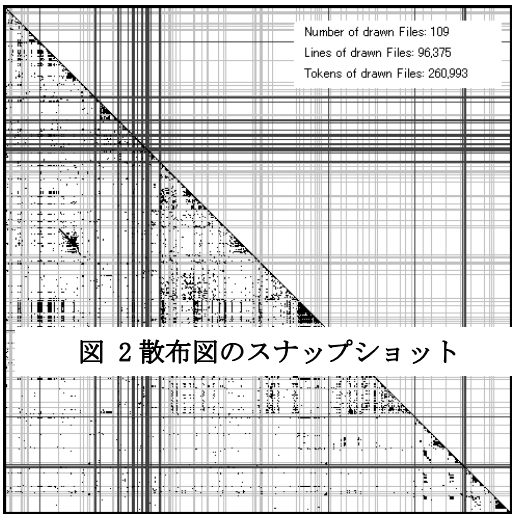

図 2 散布図のスナップショット

報告者は開発したツールを，多くの組織に対してツールを配布してきたが，その多くは試用の段階で止まっており，商用ソフトウェアの開発プロセスに組み込んで利用している組織は少ない．その理由として挙げられるのは，既存の分析技術と現場で求められている実用性との間に溝があるからである．

[1] T. Kamiya, S. Kusumoto, K. Inoue : “CCFinder: A multi-linguistic token-based code clone detection system for large scale source code”, IEEE Transactions on Software Engineering, 28(7), pp.654-670, July 2002.

[2] 肥後芳樹，吉田則裕，楠本真二，井上克郎: “産学連携に基づいたコードクローン可視化手法の提案と実装”，情報処理学会論文誌，48(2)，pp.811-822，2007 年2 月．

２．研究の目的

本研究の目的は，実際のソフトウェア開発・保守現場で使える技術としてコードクローン分析技術を確立させることである．この目的達成のためには，産業界からの意見は欠くことができない．そのため，報告者は CCFinder や Gemini を産業界に配布し，広く使っていただいている．本研究開始時までに，国内外の個人・組織を合わせて 300 近いユーザに配布し，多くのフィードバックを得ている．

３．研究の方法

本研究では，コードクローン情報を有効に用いるために，「ギャップを含むコードクローンの特定」と，「バグを含んでいる可能性の高いコードクローンの特定」に取り組む．

(1)．ギャップを含むコードクローンの特定

図 3 はコピーアンドペーストによる再利用の流れを表している．コピーアンドペースト後，まったく修正が加わらない場合は「完全一致クローン」となり，変数名や関数名などの識別子名が変更された場合は「名前変更クローン」となる．さらに，文の追加や削除，変更が行われた場合は，コピー元と対応する部分を持たない「ギャップを含むクローン」となる．このうち現在 CCFinder で検出できているのは，「完全一致クローン」と「名前変更クローン」のみであり，「ギャップを含むクローン」は検出することができない．サイズ

の小さいコードクローンも検出するように設定を行った上で CCFinder を実行すれば，ギャップの前後の部分がそれぞれ異なる「完全一致クローン」や「名前変更クローン」として出力はされるが，それを分析者が 1 つの「ギャップを含むクローン」であると認識することは難しい．しかし，実際の開発・保守作業では，このようなコピーアンドペースト後の修正はしばしば行われており，「ギャップを含むクローン」を検出することは，コードクローン分析を実用的なものにするために必要である．

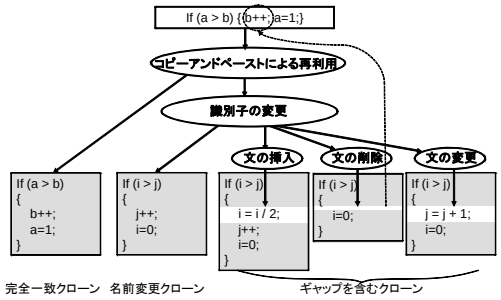

図 3 コピーアンドペーストの流れ

本研究では，多頻度グラフマイニング手法を応用することにより，ギャップを含むクローンを検出する手法を提案した．多頻度グラフマイニングとは，グラフの集合が与えられた時に，その集合の中に一定回数以上出現する部分グラフを検出する手法である[3]．全ての部分グラフを検出する場合は，NP 完全として知られる部分グラフ同形問題となるので実用的ではない．しかし，「検出する部分グラフは分岐・閉路を含まない」という条件を加えることにより，高速に検出が可能である．本研究では，下記の手順でギャップを含むコードクローンを検出する．

① CCFinder を用いて，コードクローンを検出する．
② 各コードクローンを 1 つの頂点としてグラフを構築する．各ファイルに対して 1 つのグラフが構築される．また，同一ファイル内の各コードクローン間の距離を調べ，（利用者が予め設定した）閾値よりも近い場合は，その頂点間に辺を引く．
③ 上記 2. で構築されたグラフの集合から，多頻度グラフマイニング手法を用いて 2 回以上出現する部分グラフを検出する．検出された部分グラフがギャップを含むクローンである．

[3] 猪口明博，鷲尾隆，元田浩：“多頻度グラフマイニング手法の一般化”，人工知能学会論文誌，19(5)，pp. 368-378，2004 年5 月．

(2). バグを含んでいる可能性の高いコードクローンの特定

コードクローンの存在はソフトウェア保守に悪影響を与えるが，全てのコードクローンが問題を引き起こすわけではない．そのため全てのコードクローンを調査せずに，バグを含んでいる可能性の高いコードクローンのみを調査することで作業量の削減が見込める．
これまでにさまざまな検出手法が提案されているが，各検出手法は独自にコードクローンの定義を定めているため，コードクローンの厳密な定義は存在しない．Bellon らは，各手法で異なる定義を体系的にまとめ，次の 3 つのタイプに分類している[4]．

-タイプ 1：空白やタブの有無，括弧や改行の位置などのコーディングスタイルを除いて，完全に一致するコードクローン
-タイプ 2：変数名，関数名などのユーザ定義名や変数の方などの一部の予約後の実が異なるコードクローン
- タイプ 3：タイプ 2 の変更に加えて，文の挿入，削除，変更のような更なる修正が加えられ，ギャップを含むコードクローン

本研究では，この分類をさらに細かく分けた以下の分類を提案した．

-タイプ 1a：表面上完全に同一なコードクローン
-タイプ 1b：空白やタブの有無，括弧や改行位置に違いがあるコードクローン
-タイプ 2a：タイプ 2 に属するコードクローンのうち，変数の対応が取れているもの
-タイプ 2b：タイプ 2 に属するコードクローンのうち，変数の対応が取れていないもの
-タイプ 3：文の挿入，削除，変更が行われたコードクローン

コードクローンは各タイプで特徴が異なるため，起こりうる問題もそれぞれ異なる．以下の 3 つのタイプは他のタイプに比べて問題が起こりやすいと考えられるものである．

-タイプ 1b：プログラムの意味的には同一であるが，コーディングスタイルが異なるため，見た目が異なる．従って，可読性が低い可能性が考えられる．
-タイプ 2b：変数名，関数名などのユーザ定義名や変数の型などの予約後が変更されているが，コードクローン間でそれらの対応が取れていない．したがって，識別子の修正漏れ，もしくは修正ミスの可能性が考えられる．
-タイプ 3：文の挿入，文の削除，文の変更が行われている．したがって，文単位での修正

漏れ，修正ミスの可能性が考えられる．
本研究では，このうち，とくに，タイプ 2b とタイプ 3 を検出することを目的にしている．

現在，さまざまなコードクローン検出手法があり，それらの検出能力は異なる．図 4 は手法によって検出されるコードクローンのタイプが異なることを表すモデルである．それら異なるコードクローン検出ツールの差分を取得す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．

検出手法	コードクローンタイプ				
	1a	1b	2a	2b	3
手法 A	○	×	×	×	×
手法 B	○	○	×	×	×
手法 C	○	○	○	×	×
手法 D	○	○	○	○	×
手法 E	○	○	○	○	○

図 4 検出手法と検出可能なコードクローンのモデル

例えば，手法D と手法C の出力結果の差分を取得すると，タイプ2b のコードクローンを抽出できる．同様に，手法E と手法D の出力結果の差分を取得すると，タイプ3 のコードクローンを抽出できる．提案手法は既存のコードクローン検出ツールの出力結果を用い，直接対象ソースコードの解析は行わない．そのため，差分を取得する2つのコードクローン検出ツールが対応してさえいれば，どのようなプログラミング言語に対しても適用可能である．

[4] 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, pp. 804-818, Sep 2007.

４．研究成果

(1).ギャップを含むコードクローンの特定

提案手法を実装し C 言語と Java 言語で記述されたオープンソースソフトウェアに対して実験を行った．単純に検出を行った場合では，有益と判断されたコードクローンは特定されたすべての内の 30%程度であったが，適切にフィルタリングを行うことにより，特定された約 95%が有益と判断された．

(2)バグを含んでいる可能性の高いコードクローンの特定

提案手法を実装し，Java 言語で記述されたソフトウェアに対して実験を行った．提案手法を用いない場合は（検出されたすべてのコードクローンについてバグの有無を調査した場合は），約 150 時間が必要であったのに比べ，提案手法を用いた場合は，約 8 時間で調査を完了することができた．しかし，提案手法により，バグを含むコードクローンを 1 つ誤って取り除いてしまっていたため，この点に関しては，改良を行う必要があると考えている．

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２件）

① Yoshiki Higo, Shinji Kusumoto, and Katsuro Inoue, "A Metric-based Approach to Identifying Refactoring Opportunities for Merging Code Clones in a Java Software System", Journal of Software Maintenance and Evolution: Research and Practice, Volume 20, Issue 6, pp.435-461, November, 2008.（査読あり）

② 肥後芳樹，楠本真二，井上克郎，"コードクローン検出とその関連技術"，電子情報通信学会論文誌 D， Vol.91-D， No.6，pp.1465-1481，2008 年 6 月．（査読あり）

〔学会発表〕（計７件）

①田中健介，肥後芳樹，楠本真二，"ソースコードの重複度を用いたオープンソースソフトウェアライセンス違反の検出"，情報処理学会ウインターワークショップ 2009・イン・宮崎論文集，pp.11-12，2009 年 1 月 24 日，査読あり．

② 佐野由希子，肥後芳樹，楠本真二，"クラス階層構造を利用したリファクタリング支援手法の改良"，情報処理学会関西支部 支部大会講演論文集，pp.63-66，2008 年 10 月 24 日．

③ 澤健一，肥後芳樹，楠本真二，"コードクローン検出ツールの差異情報を用いた不具合検出手法の提案と評価"，電子情報通信学会技術研究報告SS2008-24，Vol.108，No.173，pp.67-72，2008 年 8 月 1 日．

④ 宮崎宏海，肥後芳樹，井上克郎，"アイテムセットマイニングを利用したコードクローン分析作業の効率向上"，電子情報通信学会技術研究報告SS2008-18，Vol.108，No.173，

pp.31-36, 2008年7月31日.

⑤ 澤健一, 肥後芳樹, 楠本真二, ″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法″, 情報処理学会第70回全国大会講演論文集第5分冊, 2008年3月13日.

⑥ 肥後芳樹, 植田康士, 松下誠, 楠本真二, 井上克郎, ″AGMアルゴリズムを用いたギャップを含むコードクローン情報の生成″, 電子情報通信学会技術研究報告SS2007-48, Vol.107, No.392, pp.61-66, 2007年12月17日.

⑦ Yoshiki Higo, Yasushi Ueda, Shinji kusumoto, and Katsuro Inoue, ″Simultaneous Modification Support based on Code Clone Analysis″, Proc. of the 14th Asia-Pacific Software Engineering Conference (APSEC2007), pp.262-269, Nagoya, Japan, December 5th, 2007 , 査読あり.

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・助教
研究者番号：70452414