コードクローンの編集過程における開発者の潜在的社会構造に関する分析

Analyzing Latent Social Structures among Developers in Editing Code Clones

久木田 雄亮 †　大平　雅雄 †
Yusuke Kukita　Masao Ohira

1. はじめに

ソフトウェア開発・保守の過程では，生産性の向上を目的として既存コードが頻繁に再利用されることがあるため,「同一の」または「類似する」コード片（本稿ではコードクローンあるいは単にクローンと呼ぶ）がソースコード全体に遍在することが少なくない．コードクローン中の欠陥を除去するなどの理由により，コードクローンに変更を加える必要性が生じた場合，変更箇所が多岐に渡るため通常よりも保守コストが増加したり，変更漏れにより新たに不具合を混入してしまう恐れがある．そのため，コードクローンの検出手法や分析手法がこれまで盛んに研究されてきた [6]．しかしながら，既存研究の多くは主に，ある時点のソフトウェアシステムに含まれるコードクローンあるいはコードクローンの変更がソフトウェアシステムの品質に与える影響に着目しているため，コードクローンが作成される過程や原因については未だ十分には明らかになっていない．

本研究では特に，開発者間でのコードクローンの利用（共有）とソフトウェア品質との関係に着目したコードクローンの分析をおこなう．近年のソフトウェアシステムは長期に渡り保守運用されることが多く担当者が入れ替わることも少なくない．コードクローンの作成とその利用過程を開発者間の協調作業の観点から大規模な調査をおこなうことで，コードクローンの長期的かつ安全な管理方法について有益な知見を提供できる可能性がある．

続く 2 章では，コードクローンの作成および利用過程における人的影響を調査するために現在構築中のツール（CCT: Code Clones Tracer）を紹介する．3 章では，CCT を用いて今後実施する予定の分析内容について議論する．最後に 4 章で本稿のまとめと今後の課題について述べる．

2. コードクローンの作成・利用過程における人的影響を調査するための追跡ツール

CCT は，コードクローンの作成および利用過程における人的影響の調査を支援するためのツールである．数千から数万リビジョンからなるソフトウェアシステムに存在するコードクローンの追跡を想定しており，高速にクローンを特定することができる粗粒度なクローン特定方法 [5] を用いてクローンを追跡する *．CCT は，以下で説明する処理によりクローンを特定・追跡するとともに，クローン作成・利用者および不具合混入コミットを特定する．

(1) 各リビジョンのコード片特定：分析対象ファイル中に含まれるすべてのコード片を特定する．具体的には，追加・変更・削除されたファイルをすべてのリビジョンに対して特定し，ブロック単位 (クラス，メソッド，ブロック文) でコード片を特定する．最初のリビジョンを起点として，以降のリビジョンについてコード片を順次特定する．変更が加えられたファイルに対してのみコード片を特定することで，各リビジョンに含まれるすべての構成ファイル群に対してコード片を特定するよりも大幅に処理時間を短縮することができる．

(2) クローンの特定：まず，ブロックに含まれる変数名とリテラルは特殊文字にすることで正規化し，変数名やリテラルのみが異なるようなクローンを特定できるようにしておく．さらに，正規化されたブロックの文字列をもとにハッシュ値を算出する．そして，ハッシュ値の比較により，ハッシュ値の一致するブロックをクローンとして特定する．なお，クローンの特定においても，最初のリビジョンではすべてのコード片に対してクローンの特定処理をおこなうが，以降のリビジョンについては変更が加えられたファイルに含まれるコード片のみを対象にハッシュ値を求めクローンを特定する．

(3) クローンの追跡: クローン集合のリビジョン間での前後関係を特定しクローン集合をリビジョン間で追跡する．まず，最初のリビジョンに含まれるすべてのクローンのハッシュ値を比較し，同じハッシュ値を持つクローンの集合をクローン集合としてまとめる．次に，次のリビジョンで追加・変更・削除されたコード片のハッシュ値を調べ，直前のリビジョンのクローンのハッシュ値と比較をおこない，一致するクローン集合間にリンクを貼り追跡する．

(4) クローンの利用・作成に関与した開発者の特定：クローンの作成・利用過程に関与した開発者をクローン集合を含むリビジョンから特定し，クローン集合に対す

† 和歌山大学, Wakayama University

*実際，クローンの特定までの処理（コード片抽出，正規化，ハッシュ化）は，CRD[2] ベースのクローン追跡ツール ECTEC (Enhancement of CRD-based Clone Tracking)[3] の一部を再利用して実装している．

る変更操作の種類に基づいて作成者・利用者・変更者・削除者の 4 種類に分けて特定する．

(5) 不具合混入クローンの特定：(1) から (4) までは Git リポジトリから得られる情報に基づいた処理であり，クローンの作成・利用過程に関与した開発者が混入した不具合に関する情報は別途，不具合管理システム（Bugzilla や Jira など）から取得する必要がある．本ツールには，SSZ アルゴリズム [4] を用いて不具合混入コミットを特定し，不具合を混入した開発者を分析する機能を実装した．SZZ アルゴリズムによって特定した不具合混入コミットと (3) までのクローン追跡情報を照らし合わせることによって不具合が混入した開発者とクローンを特定する．

3. 開発者の潜在的社会構造とソフトウェア品質の分析

大規模 OSS 開発では依存関係のあるような特定のモジュール群を共同で作成するサブグループが形成されることが知られている．また，密に協働するサブグループが作成するモジュールは，その他のモジュールに比べて品質が高い傾向がある [1]．

CCT を用いた分析ではコードクローンの共同保守における開発者の潜在的社会構造とソフトウェア品質との関係について調査する．開発者の潜在的社会構造とは，コミュニケーションを取る回数や共同作業が多くなることによって，開発者同士の結びつきが強くなり構築されるものである．潜在的社会構造が構築されている開発者同士が共同でコードクローンの作成・変更をおこなっていれば，コードクローンの品質が高いと考える仮説を立てて分析をおこなう．

具体的には，共同作業を多くおこなっている開発者同士が作成・変更に関わっているコードクローンと，共同作業をあまりおこなっていない開発者同士で作成・変更するコードクローンにおける不具合の発生数の違いを以下の手順で分析する．まず，コードクローンの作成・変更に関わった開発者を特定する．あるコードクローンがいつ，どの開発者によって作成され，変更されたかを特定する．次に，コードクローンとコードクローンに埋め込まれている不具合の関係を特定する．そして，潜在的社会構造が構築されている開発者同士で作成・変更がおこなわれたコードクローンとそうでない開発者同士で作成・変更が行われたコードクローンの不具合発生数を比較することによってコードクローン品質への影響を分析する．

このような分析をおこなうことで，これまでの研究でおこなわれていたファイルやモジュール単位よりも粒度の細かな分析が可能となる．コードクローンの共同保守において，他者が作成したコードクローンを変更したり利用する際に留意すべき事柄などについて有益な知見を提供できる可能性がある．

4. まとめ

本稿では，コードクローンの作成および利用過程における人的影響を調査するためにツール CCT (Code Clone Tracer) を紹介した．そして，コードクローンの編集過程における開発者の潜在的社会構造とソフトウェア品質の分析方法を示した．今後は本稿で紹介した分析を実施するとともに，ツールの改良にも取り組む予定である．

謝辞

本研究の一部は，文部科学省科学研究補助金（基盤 (C): 15K00101）による助成を受けた．本稿で紹介した CCT は，ECTEC[3] の一部を再利用して実装した．

参考文献

[1] Christian Bird, David Pattison, Raissa D'Souza, Vladimir Filkov, and Premkumar Devanbu. Latent social structure in open source projects. In *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE2008)*, pp. 24–35, 2008.

[2] Ekwa Duala-Ekoko and Martin P. Robillard. Clone region descriptors: Representing and tracking duplication in source code. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, Vol. 20, No. 1, pp. 3:1–3:31, 2010.

[3] Yoshiki Higo, Keisuke Hotta, and Shinji Kusumoto. Enhancement of crd-based clone tracking. In *Proceedings of the 2013 International Workshop on Principles of Software Evolution (IWPSE2013)*, pp. 28–37, 2013.

[4] Jacek Śliwerski, Thomas Zimmermann, and Andreas Zeller. When do changes induce fixes? In *Proceedings of the 2005 International Workshop on Mining Software Repositories (MSR2005)*, pp. 1–5, 2005.

[5] 堀田圭佑, 楊嘉晨, 肥後芳樹, 楠本真二. 粗粒度なコードクローン検出手法の精度に関する調査. 情報処理学会論文誌, Vol. 56, No. 2, pp. 580–592, feb 2015.

[6] 堀田圭佑, 肥後芳樹, 楠本真二. 生成抑止，分析効率化，不具合検出を中心としたコードクローン管理支援技術に関する研究動向. コンピュータソフトウェア, Vol. 31, No. 1, pp. 14–29, 2 2014.